Für
CPC 464/664/6128

Das Schneider CPC Grafikbuch

SYBEX



Hans Lorenz Schneider

Das Schneider CPC
Grafikbuch

Das Schneider CPC Grafikbuch

Hans Lorenz Schneider



Umschlagentwurf: Daniel Boucherie/tgr
Satz: tgr – typo-grafik-repro gmbh, Remscheid
Gesamtherstellung: Boss-Druck und Verlag, Kleve

ISBN 3-88745-611-4
1. Auflage 1986

Printed in Germany

Inhaltsverzeichnis

Kapitel 3

Kapitel 4

Vorwort

Die Schneider CPC-Rechner weisen gegenüber dem bisherigen „Standard" bei Home-Computern insbesondere im Bereich der Grafik einige wesentliche Vorzüge auf. Mit dem vorliegenden Buch haben wir versucht, den vielfältigen Anwendungsmöglichkeiten von Grafiken Rechnung zu tragen. Dabei soll das Buch sowohl dem Anfänger den Einstieg ermöglichen als auch dem Profi Tips zur Verfeinerung seiner Programme geben.

Besonderer Wert wurde auf die Beschreibung des Handwerkszeugs gelegt, d. h. auf die Grafikmöglichkeiten des CPC. Das erste Kapitel sollte von jedem Leser bei Bedarf auszugsweise gelesen werden. Nicht jeden werden die Betriebssystem-Routinen interessieren, sofern er nicht mit Maschinensprache-Programmen arbeitet. Aus diesem Grunde wurde auch Kapitel 2 zweigeteilt, um sowohl „nur BASIC-Programmierern" als auch Maschinensprache-Spezialisten die Möglichkeit der Ergänzung der Grafikbefehle zu geben. Die weiteren Kapitel sind dann verschiedenen Anwendungsmöglichkeiten der hochauflösenden Grafik gewidmet.

Besonders bedanken möchte ich mich bei Reinhard Damm, der das Kapitel über dreidimensionale Grafiken ausgearbeitet hat, und Andreas Kißlinger, der mir bei der Erstellung der Befehlserweiterung in Maschinensprache, den Sprites und der Hardcopy behilflich war.

München, im Januar 1986 — Hans Lorenz Schneider

Einleitung

Wie im Vorwort erwähnt, soll das Buch sowohl Anfängern als auch Fortgeschrittenen und Profis gerecht werden. Dazu beschäftigt sich Kapitel 1 mit den durch den CPC 464, CPC 664 und CPC 6128 gegebenen Grafikgrundlagen.

Obwohl die CPC-Rechner viele Grafikbefehle zur Verfügung stellen, haben wir für Sie doch einige Befehlserweiterungen in Kapitel 2 – sowohl in BASIC als auch in Maschinensprache – vorgestellt, die bei häufiger Grafikanwendung von Vorteil sein können. Eine Anwendung der BASIC-Unterprogramme finden Sie in Kapitel 3, wo ein Programm aufgezeigt wird, mit dem Sie Grafiken unter Zuhilfenahme des Joysticks entwerfen können. Auch Kapitel 4 (Diagramme) stellte eine Anwendung der in Kapitel 2 vorgestellten Befehlserweiterung dar.

Wer weniger technisch, sondern mehr gestalterisch tätig sein will, findet in Kapitel 5 eine grundlegende Einführung, wobei auch die wichtigen Winkelfunktionen Sinus und Cosinus ausführlich besprochen werden. Tricks mit laufenden Farben, der Transparent-Modus und die verschiedenen Schreib-Modi beleben sicherlich hübsche Grafiken zusätzlich.

Wenn jemand den mathematischen Aufwand nicht scheut, wird er durch die Lektüre des Kapitels 6 zur Gestaltung räumlicher Grafiken aufgefordert. Sogar Stereobilder – bei Verwendung einer entsprechenden Rot-Grün-Brille – können entworfen werden. Da bei der Erzeugung von dreidimensionalen Funktionen ein sehr hoher Anteil an Berechnungen auftritt, ist es sinnvoll, öfter zu verwendende Darstellungen von dreidimensionalen Funktionen auf Diskette oder Kassette abzulegen. Hier ist Kapitel 7 sehr hilfreich. Die Ausgabe einer Grafik auf den Drucker wird durch das in Kapitel 9 vorgestellte Hardcopy-Programm ermöglicht.

Da die CPC-Rechner von Hause aus keine Sprites – frei bewegbare Bildschirmausschnitte – kennen, wird in Kapitel 8 ein Weg für ihre Darstellung aufgezeigt.

Und nun viel Spaß beim Lesen, Programmieren und besonders beim Erstellen von Grafiken.

Kapitel 1
Die Grafikeigenschaften der CPC-Rechner

Bevor wir in den weiteren Kapiteln Erweiterungen der Grafikbefehle sowie verschiedene Anwendungen der Grafikmöglichkeiten beschreiben, wollen wir zuerst die grundlegenden Eigenschaften des CPC 464, CPC 664 und CPC 6128 im Hinblick auf die Grafik beleuchten. Dazu wird zunächst die benötigte Hardware vorgestellt und der Weg eines Bildpunktes bis zum Bildschirm aufgezeigt.

Besonders für Maschinensprache-Programmierer dürfte die Aufstellung der Betriebssystem-Routinen (Grafik) von Interesse sein. Weitere Betriebssystem-Routinen sind in Kapitel 2 beschrieben (in der Hauptsache Fließkommaoperationen). Als nächstes wird der Grafikaufbau auf dem Bildschirm besprochen sowie auf die vorhandenen Grafikbefehle kurz eingegangen.

Neben der hochauflösenden Grafik weisen die CPC-Rechner durch ihren umfangreichen Zeichensatz auch einige Grafikelemente innerhalb desselben auf. Eine Übersicht und ein paar Hinweise für das Arbeiten mit diesen Grafikzeichen bilden den Abschluß dieses Einführungskapitels.

1.1 HARDWARE-AUFBAU: DER WEG EINES BILDPUNKTES

Die Grafik des CPC wird im wesentlichen von vier Hardware-Komponenten bestimmt. Diese wiederum übernehmen ganz unterschiedliche Aufgaben.

Der Mikroprozessor, ein Z80-A, ist für das Beschreiben des Bildschirmspeichers zuständig. Alle Grafik-Routinen und Zeichenausgabe-Routinen unterstehen seinem Kommando. Eine nähere Beschreibung der Routinen erfolgt im Abschnitt „Betriebssystem-Routinen“.

Die zweite Komponente ist der Bildspeicher. Er ist als Teil des Arbeitsspeichers ausgelegt. Normalerweise liegt der Bildspeicher zwischen &C000 und &FFFF und ist damit 16 KByte lang; für jeden Bildschirmpunkt wird mindestens ein Bit benötigt (Bank 3). Der Bildspeicher läßt sich für besondere An-

wendungen, wie z. B. einen schnellen Bildwechsel, durch eine Routine auf einen anderen 16-KByte-Bereich festlegen. Sinnvoll ist jedoch fast immer der Standardbereich, da in den Bänken 0 und 2 Teile des Betriebssystems liegen. Die Verwendung der Bank 1, dem Bereich von &4000 bis &7FFF, ist möglich, aber der zur Verfügung stehende BASIC-Speicher wird dadurch auf etwa 16 KByte eingeschränkt. Eine sinnvolle Verlegung des Bildspeichers auf eine andere Bank bringt daher nur bei Assemblerprogrammierung etwas, da Maschinenprogramme auch aufgeteilt im Speicher stehen können.

Etwas anderes ist ebenfalls noch erwähnenswert: Wenn man nämlich die Breite einer Bildzeile, also 80 Byte, mit der Bildzeilenzahl 200 multipliziert, so kommt man auf einen Wert von 16000 Byte. Der Bildspeicher ist aber 16 KByte (also 16384 Byte) lang. Wo sind also diese 384 Bytes geblieben? Des Rätsels Lösung ist folgende: Es werden nämlich jeweils die letzten 48 Byte eines jeden 2-KByte-Blocks von der Bilderzeugung nicht benutzt.

Nach einem MODE-Befehl steht die Bildspeicheradresse immer am Anfang, also bei &C000; bei jedem Scrollen ändert sie sich um &50. Die jetzt angegebenen Adressen gelten nur nach einem MODE-Befehl, sie müssen je nach Anfangsadresse des Bildspeichers korrigiert werden. Die Adressen der ungenutzten Speicherzellen sind &C7D0 bis &C7FF und alle die durch Addition von &800 entstehen. Diese Speicherzellen können bis zu einem Scroll, einem CLS-, einem CLG- oder einem MODE-Befehl zur Ablage eines kurzen Maschinenprogramms oder von Variablen desselben benutzt werden.

Die aktuelle Anfangsadresse des ersten 2-KByte-Blocks kann durch eine Maschinen-Routine erhalten werden. Die Routine &BC0B gibt dazu im Akku die Basisadresse, also &00, &40, &80 oder (meistens) &C0 zurück. Das HL-Register enthält die zugehörige Distanz des ersten 2-KByte-Blocks. Nach einem MODE-Befehl liefert die Routine also A=&C0 und HL=&0000 zurück. Aus diesen Daten kann man dann den ungenutzten Speicherplatz berechnen.

Die Speicher-ICs des CPC sind vom Typ 4164, also dynamische Speicherchips. Bei dieser Art von Speichern muß in bestimmten Zeitabständen die Speicherinformation aufgefrischt werden, da sie sonst verlorengeht. Normalerweise macht dies ein spezieller Schaltkreis oder der Mikroprozessor. Beim CPC ist man da einen etwas anderen Weg gegangen.

Dies führt zur dritten Komponente, dem Video-Controller, oft auch als CRT- oder Bildschirm-Controller bezeichnet. Dieser Controller greift in regelmäßigen Zeitabständen auf den Bildspeicher zu und gibt die Bildinformation weiter. Die beim Zugriff ausgegebenen Adressen werden durch einen Schaltungstrick zum Refresh des Arbeitsspeichers genutzt. Deshalb wird nach einem Reset als erstes IC der Controller initialisiert. Erst nach dieser Einstellung greift

er periodisch auf den Speicher zu und ermöglicht es dem Prozessor, Daten im Arbeitsspeicher abzulegen, ohne daß sie wieder verschwinden.

Weiterhin übernimmt er die Erzeugung der – für ein Video-Bild sehr wichtigen – Synchronsignale. Der Controller bestimmt die Anzahl der Zeichen bzw. Bildpunkte am Bildschirm. Durch eine geeignete Programmierung des Controllers kann z. B. die Bildbreite und -höhe eingeschränkt oder vergrößert werden.

Der Video-Controller ist vom Typ 6845. Er kann bis zu 512 KByte Bildspeicher verwalten. Der Zugriff des Controllers auf den Speicher ist so organisiert, daß ein sehr einfaches Scrollen des ganzen Bildes möglich ist. Beim Scrollen des gesamten Bildes wird nämlich dem Controller nur eine um 80 Byte veränderte Anfangsadresse mitgeteilt. Er hat 18 Register, die – größtenteils – für eine Bilddarstellung programmiert werden müssen. Will man die Register selbst ansprechen, so kann dies unter BASIC mit dem Befehl

```
OUT &BCxx,nn : OUT &BDxx,mm
```

geschehen, wobei xx eine beliebige HEX-Zahl, z. B. 00, sein kann. Der erste der beiden OUT-Befehle wählt das Register nn des CRT-Controllers an. Der zweite schreibt die Daten mm in das gewählte Register. Wenn man die Register beschreibt, ohne sich vorher über die Auswirkungen im klaren zu sein, so kann es vorkommen, daß man einen Reset durchführen muß, um wieder weiterarbeiten zu können. Daher ist es sinnvoll, vorher erzeugte Programme erst einmal zu speichern. Der Video-Controller kann auch einen Cursor erzeugen und die Abfrage eines Lightpens übernehmen. Diese Optionen werden jedoch vom Aufbau des CPC noch nicht genutzt. Die softwaremäßige Cursor-Erzeugung beim CPC hat auch den Vorteil, daß die Form des Cursors nicht festgelegt ist, sondern man kann über die Verwendung der Sprungtabelle mit einer eigenen Routine einen Cursor erzeugen. Dieser eigene Cursor kann dann z. B. blinken und aussehen wie der Unterstrich.

Die Register des Controllers haben folgende Aufgaben:

Register-Nr.	Bedeutung	Standardwert (dezimal)
0	Gesamtzahl Zeichen horizontal	63
1	dargestellte Zeichen horizontal	40
2	horizontale Synchronisationsposition, Zeichen	46
3	Breite der Synchronsignale	&X10001110
4	Gesamtzahl Zeichenzeilen vertikal	38

Register-Nr.	Bedeutung	Standardwert (dezimal)
5	Abgleich vertikal in Bildzeilen	0
6	dargestellte Zeilen vertikal	25
7	vertikale Synchronisationsposition, Zeile	30
8	Zeilensprung und doppelte Zeichendichte	0
9	Bildzeilen – 1 pro Zeichenzeile	7
10	Cursor-Anfangsbildzeile	nicht benutzt
11	Cursor-Endbildzeile und Blinkmodus	nicht benutzt
12	Speicheranfangsadresse (High Byte)	&30 je nach
13	Speicheranfangsadresse (Low Byte)	&00 Scroll
14	Cursor-Adresse (High Byte)	nicht benutzt
15	Cursor-Adresse (Low Byte)	nicht benutzt
16	Lightpen-Adresse (High Byte)	nicht benutzt
17	Lightpen-Adresse (Low Byte)	nicht benutzt

Bemerkung zum Register 3: In diesem Register wird die Dauer des vertikalen und des horizontalen Synchronsignals eingestellt. Die Bits 0 bis 3 sind für das horizontale und die anderen Bits für das vertikale Synchronsignal zuständig. Die Standardwerte sind 8 für die vertikale und 14 für die horizontale Synchronisation.

Die Register 0 bis 13 können mit den oben angegebenen OUT-Befehlen nur beschrieben werden. Die Register 16 bis 17 können nur gelesen und die Register 14 und 15 sowohl gelesen als auch beschrieben werden.

Dazu ist mit

```
OUT &BCxx,nn : Register = INP(&BFxx)
```

der Inhalt des Registers nn in die Variable „Register“ einzulesen. Der Wert xx kann wieder beliebig angenommen werden.

Die Register 10, 11 und 14 bis 17 werden vom Betriebssystem des CPC noch nicht unterstützt. Das Cursor-Signal und der Eingang für den Lichtgriffel sind aber am Systembus herausgeführt.

Der vierte Teil der Bilderzeugung wird schließlich vom Gate-Array des CPC übernommen. Dieses IC ist für die Synchronisation des Mikroprozessors und des Video-Controllers zuständig. Diese Verschaltung von CPU und CRT-Controller wird vorgenommen, damit der Mikroprozessor nicht etwa den Bildspeicher beschreibt, wenn zur selben Zeit gerade der Video-Controller ein Byte lesen will. In diesem Fall hat die Bilderzeugung den Vorrang, damit das Monitorbild nicht in diesem Moment flackert. Der Prozessor bekommt

während dieser Zeit ein WAIT-Signal. Die Synchronisation erfolgt so, daß CPU und CRT-Controller immer abwechselnd auf den Speicher zugreifen. Die zweite wichtige Aufgabe, die das Gate-Array übernimmt, ist die Umschlüsselung des vom CRT-Controller erhaltenen Bytes in ein RGB-Farbsignal.

Die Erzeugung eines Bildes durch diese vier Komponenten geschieht in mehreren Schritten. Zuerst wird eine Information an die Betriebssystem-Routinen übergeben. Ist diese z. B. der Zeichenbefehl PLOT, so schreibt der Prozessor einen Punkt, nachdem er die Korrektheit der Lage überprüft und die Adresse im Speicher berechnet hat, als Zahlenwert in den Bildspeicher. Bei diesem Vorgang müssen eventuell vorhandene Bildpunkte berücksichtigt werden. Am einfachsten ist dies im MODE 2, dann entspricht nämlich ein gesetztes Bit einem Bildpunkt der Farbe Eins. In den anderen Modi muß die Farbinformation codiert und eingeschrieben werden. Die Codierung in den verschiedenen Bildschirmmodi ist ziemlich kompliziert. Die Verwendung der einzelnen Bits in einem Byte geht nicht – wie man annehmen könnte – der Reihenfolge der Pixel nach, sondern alterniert.

Die Verwendung der Bits ist in der folgenden Tabelle veranschaulicht. Dabei bedeutet Pn den beschriebenen Pixel und Bn die Wertigkeit dieses Bits in dem angegebenen Pixel. Die Pixels sind – im Gegensatz zu den Bits – in aufsteigender Reihenfolge angegeben, d. h. P0 ist ganz links, und alle anderen Pixels befinden sich rechts davon.

Bedeutung der Bits eines Bytes im Bildspeicher:

MODE	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	P0 B0	P1 B0	P0 B2	P1 B2	P0 B1	P1 B1	P0 B3	P1 B3
1	P0 B0	P1 B0	P2 B0	P3 B0	P0 B1	P1 B1	P2 B1	P3 B1
2	P0 B0	P1 B0	P2 B0	P3 B0	P4 B0	P5 B0	P6 B0	P7 B0

So steht die Bildinformation im Speicher. Die für einen Pixel jeweils zusammengehörenden Bits stellen ein Wort dar, in dem die der jeweiligen Farbe entsprechende Zahl als Binärwort gespeichert ist.

Der nächste Schritt wird vom Bildschirm-Controller durchgeführt. Er liest Byte für Byte aus dem Bildspeicher und gibt in einprogrammierten Zeitabständen ein Signal für die horizontale und die vertikale Synchronisation ab. Die horizontale Synchronisation dient dazu, die Bildzeilen genau untereinander zu positionieren. Die vertikale Synchronisation gibt dem Monitor zu verstehen, wann ein neuer Bildaufbau erfolgt, wann also der Schreibstrahl des Monitors nach links oben springen soll. Die ausgelesenen Bytes gibt der CRT-Controller an das Gate-Array weiter.

Das Gate-Array sorgt im Teil der computerinternen Bilderzeugung dafür, daß das vom Controller übertragene Byte in eine Farbinformation aufgeschlüsselt wird. Dazu übersetzt es die Farbnummer des Pixels in eine Farbhardwarenummer und gibt die dieser Nummer entsprechenden RGB-Werte aus (RGB ist die Abkürzung für Rot-Grün-Blau).

Diese Farbsignale werden noch mit den Synchronsignalen gemischt und dann als RGB-Signal an den Monitor ausgegeben. Dieser ganze Vorgang hört sich sehr einfach an. Es ist aber in Wirklichkeit ein nicht zu unterschätzender Aufwand von Elektronik damit verbunden, mit dem jedoch der normale Computer-Anwender nicht belastet wird.

1.2 DIE GRAFIK-ROUTINEN IM ROM

Adresse:		**&BBBA**
Bezeichnung englisch		GRA INITIALISE
Bezeichnung deutsch		Grafik initialisieren.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört

Bemerkung

- Setzen der Verzweigungspunkte GRA PLOT, GRA TEST und GR LINE auf ihre Standardwerte.
- Hintergrundfarbe auf INK 0 schalten.
- Grafikstift auf INK 1 schalten.
- Koordinatenursprung des Anwenders in die linke untere Ecke des Bildschirms setzen.
- Position des Grafik-Cursors auf Koordinatenursprung setzen.
- Grafikfenster auf den gesamten Bildschirm ausdehnen.

Achtung: Das Grafikfenster wird nicht gelöscht.

Adresse:		**&BBBD**
Bezeichnung englisch		GRA RESET
Bezeichnung deutsch		Grafik zurücksetzen.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Keine anderen Aktionen.

Adresse:		**&BBC0**
Bezeichnung englisch		GRA MOVE ABSOLUT
Bezeichnung deutsch		Setze den Grafik-Cursor an die angegebene Position.
Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate
	HL	Y-Koordinate
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Wie vom Arbeiten mit den Befehlen PLOT, TEST und DRAW bekannt ist, benutzen diese Befehle ebenfalls die vorliegende Routine.

Adresse:		**&BBC3**
Bezeichnung englisch		GRA MOVE RELATIVE
Bezeichnung deutsch		Grafik-Cursor relativ zur aktuellen Position setzen.

Eingabeparameter	AF	–
	BC	–
	DE	Wert auf X-Achse (mit Vorzeichen)
	HL	Wert auf Y-Achse (mit Vorzeichen)
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Die neue Position muß nicht innerhalb des definierten Grafikfensters liegen. Diese Routine wird auch von den Befehlen PLOTR, TESTR und DRAWR benutzt.

Adresse: **&BBC6**

Bezeichnung englisch		GRA ASK CURSOR
Bezeichnung deutsch		Aktuelle Position des Grafik-Cursors feststellen.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	–
	DE	X-Koordinate
	HL	Y-Koordinate
Bemerkung		Die Angabe erfolgt relativ zum definierten Koordinatenursprung. In der Regel wird die zuletzt benutzte Position des Grafik-Cursors ausgegeben.

Adresse: **&BBC9**

Bezeichnung englisch		GRA SET ORIGIN
Bezeichnung deutsch		Koordinaten des Anwenderursprungs festlegen.

Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate
	HL	Y-Koordinate
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört

Bemerkung

Die Angaben zur Positionierung des Koordinatenursprungs müssen relativ zur linken unteren Ecke des Bildschirms (Position: 0,0) angegeben werden. Die Voreinstellung ist auch die linke untere Ecke des Bildschirms (0,0). Wenn ein neuer Bildschirmmodus gesetzt ist, wird der Ursprung des Koordinatensystems automatisch auf die Standardposition gesetzt.

Adresse: **&BBCC**

Bezeichnung englisch: GRA GET ORIGIN

Bezeichnung deutsch: Ausgabe der Koordinaten des Koordinatenursprungs.

Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	–
	BC	–
	DE	X-Koordinate des Ursprungs
	HL	Y-Koordinate des Ursprungs

Bemerkung

Die Position wird relativ zur unteren linken Ecke des Bildschirms angegeben (0,0).

Adresse: **&BBCF**

Bezeichnung englisch: GRA WIN WIDTH

Bezeichnung deutsch		Linken und rechten Rand des Grafikfensters setzen.
Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate des einen Randes bezüglich des linken unteren Bildschirmrandes
	HL	X-Koordinate des anderen Randes bezüglich des linken unteren Bildschirmrandes
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Die Koordinaten werden bezüglich der linken unteren Ecke des Bildschirms (0,0) angegeben, wobei die Koordinaten als 16-Bit-Zahl mit Vorzeichen vorliegen müssen.

Der kleinere Wert aus den Registern DE und HL wird automatisch als linke Seite des Fensters angenommen, der größere Wert gibt dementsprechend den rechten Rand an, wobei die angegebenen Punkte innerhalb des Fensters liegen. Gegebenenfalls wird eine Anpassung an den Bildschirmrand vorgenommen.

Außerdem werden die Begrenzungen des Bildschirmfensters so modifiziert, daß immer ein ganzes Byte der Bildschirmdarstellung innerhalb des Fensters liegt.

Standardmäßig wird der gesamte Bildschirm mit dem Grafikfenster belegt. Die Standardgröße wird auch immer eingeschaltet, wenn der Bildschirmmodus verändert wird.

Wichtig: Obwohl bei der Verwendung der Grafikbefehle PLOT, PLOTR, DRAW und DRAWR Punkte außerhalb des Fensters angegeben werden können, werden diese aber nicht auf dem Bildschirm ausgegeben.

Adresse: **&BBD2**

Bezeichnung englisch	GRA WIN HEIGHT
Bezeichnung deutsch	Oberen und unteren Rand des Grafikfensters festlegen.

Eingabeparameter	AF	–
	BC	–
	DE	Y-Koordinate für den einen Rand
	HL	Y-Koordinate für den anderen Rand
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Siehe &BBCF. Die angegebenen Daten gelten analog.

Adresse: **&BBD5**

Bezeichnung englisch		GRA GET W WIDTH
Bezeichnung deutsch		Rechten und linken Rand des Grafikfensters holen.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	–
	DE	X-Koordinate des linken Randes
	HL	X-Koordinate des rechten Randes
Bemerkung		Die Koordinatenangabe bezieht sich auf den linken unteren Bildschirmrand (0,0). Da die Werte für die Fenster vom Rechner selbständig verändert werden können (siehe &BBCF sowie &BBD2), muß die Ausgabe nicht den gesetzten Werten entsprechen.

Adresse: **&BBD8**

Bezeichnung englisch		GRA GET W HEIGHT
Bezeichnung deutsch		Oberen und unteren Rand des Grafikfensters ausgeben.

Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	–
	DE	Y-Koordinate des oberen Randes
	HL	Y-Koordinate des unteren Randes
Bemerkung		Siehe &BBD5.

Adresse:		**&BBDB**
Bezeichnung englisch		GRA CLEAR WINDOW
Bezeichnung deutsch		Grafikfenster löschen.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Neben dem Löschen des Grafikfensters wird der Grafik-Cursor auf den Ursprung der Anwender-Koordinaten gesetzt.

Adresse:		**&BBDE**
Bezeichnung englisch		GRA SET PEN
Bezeichnung deutsch		Farbe für Grafikstift festlegen.
Eingabeparameter	A	gewünschte Farbe
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	–

	DE	–
	HL	–
Bemerkung		Die im Register A angegebene Farbe wird gegebenenfalls modifiziert, um sie dem entsprechenden Bildschirmmodus anzupassen. Der Wert wird also modulo 16 (Modus 0), modulo 4 (Modus 1) und modulo 2 (Modus 2) genommen.

Adresse:		**&BBE1**
Bezeichnung englisch		GRA GET PEN
Bezeichnung deutsch		Aktuelle Farbe des Grafikstiftes holen.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	A	Farbe des Grafikstiftes
	BC	–
	DE	–
	HL	–
Bemerkung		Alle Flags werden zerstört.

Adresse:		**&BBE4**
Bezeichnung englisch		GRA SET PAPER
Bezeichnung deutsch		Farbe des Grafikhintergrundes festlegen.
Eingabeparameter	A	Farbe
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	–
	DE	–
	HL	–
Bemerkung		Alle Flags werden zerstört.

Adresse:		**&BBE7**
Bezeichnung englisch		GRA GET PAPER
Bezeichnung deutsch		Aktuelle Farbe des Hintergrundes bestimmen.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	A	Hintergrundfarbe
	BC	–
	DE	–
	HL	–
Bemerkung		Alle Flags werden zerstört.

Adresse:		**&BBEA**
Bezeichnung englisch		GRA PLOT ABSOLUTE
Bezeichnung deutsch		Punkt an der angegebenen Position ausgeben.
Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate
	HL	Y-Koordinate
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Wie üblich wird auch der Grafik-Cursor an die Stelle des ausgegebenen Punktes bewegt. Liegt der Punkt außerhalb des aktuellen Grafikfensters, so wird er nicht ausgegeben. Die Angabe der Koordinaten erfolgt in bezug auf das vom Anwender festgelegte Koordinatensystem. Die indirekten Verzweigungspunkte GRA PLOT und GRA WRITE werden benutzt.

Adresse:		**&BBED**
Bezeichnung englisch		GRA PLOT RELATIVE
Bezeichnung deutsch		Punkt relativ zur aktuellen Position des Grafik-Cursors ausgeben.
Eingabeparameter	AF	–
	BC	–
	DE	Wert und Vorzeichen auf X-Achse
	HL	Wert und Vorzeichen auf Y-Achse
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Die Angabe der Position erfolgt relativ zu den aktuellen Koordinaten. Siehe auch &BBEA.

Adresse:		**&BBF0**
Bezeichnung englisch		GRA TEST ABSOLUTE
Bezeichnung deutsch		Punkt an angegebener Position prüfen.
Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate
	HL	Y-Koordinate
Ausgabeparameter	A	Farbauswahlnummer des Punktes
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Die Flags sind zerstört. Soll ein Punkt außerhalb des Grafikfensters getestet werden, so erfolgt keine Ausgabe. Die Angabe der Koordinaten bezieht sich auf das vom Anwender definierte Koordinatensystem. Die indirekten Verzweigungspunkte GRA TEST und SCR READ werden verwendet.

Adresse:		**&BBF3**
Bezeichnung englisch		GRA TEST RELATIVE
Bezeichnung deutsch		Farbe eines Punktes relativ zur aktuellen Cursor-Position feststellen.
Eingabeparameter	AF	–
	BC	–
	DE	Wert und Vorzeichen auf der X-Achse
	HL	Wert und Vorzeichen auf der Y-Achse
Ausgabeparameter	A	Farbauswahlnummer des gewünschten Punktes
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Die Flags werden zerstört. Siehe auch &BBF0.

Adresse:		**&BBF6**
Bezeichnung englisch		GRA LINE ABSOLUTE
Bezeichnung deutsch		Linie zur angegebenen Position ziehen.
Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate
	HL	Y-Koordinate
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Der Grafik-Cursor muß sich an einem Endpunkt der Linie befinden. Die anderen Koordinaten sind in den Registern DE und HL zu übergeben. Die Position wird in bezug auf die Anwender-Koordinaten angegeben. Verwendete indirekte Verzweigungspunkte sind: GRA LINE und SCR WRITE. Die Linie wird nur innerhalb des Grafikfensters gezogen.

Adresse:		**&BBF9**
Bezeichnung englisch		GRA LINE RELATIVE
Bezeichnung deutsch		Linie mit relativer Positionsangabe ziehen.
Eingabeparameter	AF	–
	BC	–
	DE	Wert mit Vorzeichen auf der X-Achse
	HL	Wert mit Vorzeichen auf der Y-Achse
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Die Angabe der Koordinaten bezieht sich auf die aktuelle Position des Grafik-Cursors. Verwendete indirekte Verzweigungspunkte sind: GRA LINE und SCR WRITE. Die Linie wird mit der aktuellen Grafikstiftfarbe gezogen.

Adresse:		**&BBFC**
Bezeichnung englisch		GRA WR CHAR
Bezeichnung deutsch		Zeichen auf dem Bildschirm ausgeben.
Eingabeparameter	A	auszugebendes Zeichen
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Die linke obere Ecke des Zeichens wird an der Position des Grafik-Cursors ausgegeben. Auch die Zeichen 0 bis 31 sind darstellbar. Der Grafik-Cursor wird anschließend um die Breite des Zeichens nach rechts verlegt, d. h. 32 Punkte (Modus 0), 16 Punkte (Modus 1) und 8 Punkte (Modus 2). Das Zeichen wird mit der Farbe des Grafikstiftes dargestellt.

		Als Hintergrund dient ebenfalls die Hintergrundfarbe der Grafik. Achtung: Der Hintergrund wird mit ausgegeben, so daß eine Grafik in diesem Bereich zerstört wird. Bereiche außerhalb des Grafikfensters werden nicht ausgegeben.

Adresse:		**&BB63**
Bezeichnung englisch		TXT SET GRAPHIC
Bezeichnung deutsch		Positionierung der Textausgabe im Grafikbildschirm ein- oder ausschalten.
Eingabeparameter	A	0 – ausgeschaltet sonst – angeschaltet
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	–
	DE	–
	HL	–
Bemerkung		Entspricht den Befehlen TAG und TAGOFF.

Adresse:		**&BC1D**
Bezeichnung englisch		SCR DOT POSITION
Bezeichnung deutsch		Punktmaske abfragen.
Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate im Bereich 0 – 160, 0 – 320 oder 0 – 640, je nach gewähltem Modus
	HL	Y-Koordinate im Bereich 0 – 199
Ausgabeparameter	AF	zerstört
	B	Anzahl der Bildpunkte im Byte –1
	C	Maske für den Bildpunkt
	DE	zerstört
	HL	Bildschirmadresse des Bildpunktes

Bemerkung		Diese Routine ist sinnvoll, wenn auf dem Bildschirm unter Berücksichtigung vorhandener Bildpunkte geschrieben werden soll.

Adresse:		**&BC5C**
Bezeichnung englisch		SCR PIXELS
Bezeichnung deutsch		Bildpunkt ohne Berücksichtigung des Schreib-Modus setzen.
Eingabeparameter	AF	–
	B	codierte Farbauswahlnummer
	C	Maske der/des Bildpunkte(s)
	DE	–
	HL	Bildschirmadresse der/des Bildpunkte(s)
Ausgabeparameter	AS	zerstört
	BC	–
	DE	–
	HL	–
Bemerkung		Die Anwendung dieser Routine ist sinnvoll, wenn mit Bildschirmadressen gearbeitet wird, d. h. mehrere Bildpunkte, die innerhalb eines Bytes liegen, zusammengefaßt dargestellt werden sollen. Dies kann z. B. von Vorteil sein, wenn man eine Linie ziehen will und man mit der vorliegenden Routine nicht die einzelnen Bildpunkte ansprechen muß. Der eingeschaltete Schreib-Modus (im Handbuch des 664 und 6128 auch als INK-Modus bezeichnet) bleibt dabei unberücksichtigt.

Adresse:		**&BC5F**
Bezeichnung englisch		SCR HORIZONTAL
Bezeichnung deutsch		Waagerechte Linie zeichnen.
Eingabeparameter	A	Farbe
	DE	X-Koordinate im Bereich 0 – 160, 0 – 320

		oder 0 – 640, je nach gewähltem Modus (Anfang der Linie)
	BC	X-Koordinate, Bereich wie bei DE (Ende der Linie)
	HL	Y-Koordinate im Bereich 0 – 199
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		In bestimmten Fällen ist es sinnvoll, bei zeitkritischen Grafikausgaben die Linie nicht mit dem DRAW-Befehl zu ziehen, sondern auf die vorliegende Routine zurückzugreifen. Beachten Sie, daß die Anfangskoordinate kleiner oder gleich der Endkoordinate der Linie sein muß, da keine automatische Anpassung erfolgt.

Adresse:		**&BC62**
Bezeichnung englisch		SCR VERTICAL
Bezeichnung deutsch		Senkrechte Linie ziehen.
Eingabeparameter	A	Farbe
	DE	X-Koordinate im Bereich 0 – 160, 0 – 320 oder 0 – 640, je nach Modus
	HL	Y-Koordinate im Bereich 0 – 199 (Anfang der Linie)
	BC	Y-Koordinate im Bereich 0 – 199 (Ende der Linie)
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Siehe SCR HORIZONTAL.

Zum Schluß folgen noch ein paar Grafik-Routinen, die nur für die beiden neuen Schneider-Computer gelten. Diese Routinen sind also nur auf den CPC 664 und CPC 6128 verfügbar.

Adresse:		**&BD43**
Bezeichnung englisch		(GRA INIT EXTRA)
Bezeichnung deutsch		Grafikzusatzfunktionen zurücksetzen.
Eingabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	zerstört
	BC	–
	DE	–
	HL	zerstört
Bemerkung		Setzt die Linienmaske auf 255, schaltet den Transparent-Modus für die Grafikausgabe von Zeichen aus, stellt den Verknüpfungsmodus der Grafik auf überschreibend und gibt an, daß der erste Punkt einer Linie gezeichnet werden soll.

Adresse:		**&BD46**
Bezeichnung englisch		(GRA SET BACK)
Bezeichnung deutsch		Zulassen oder Sperren der Hintergrunddarstellung bei der Ausgabe von Zeichen auf Grafikposition.
Eingabeparameter	AF	Wenn der Hintergrund dargestellt werden soll (undurchsichtig): A muß 0 enthalten Wenn der Hintergrund nicht dargestellt werden soll (durchsichtig): A muß ungleich 0 sein
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	–
	BC	–
	DE	–
	HL	–

Bemerkung		Die Darstellung des Hintergrundes eines Zeichens gilt nur für die Grafikausgabe. Die transparente Textausgabe ist gesondert anzusprechen.

Adresse:		**&BD49**
Bezeichnung englisch		(GRA SET FIRST)
Bezeichnung deutsch		Zulassen oder Sperren des ersten Punktes bei dem Zeichnen einer Linie.
Eingabeparameter	AF	Wenn der erste Punkt einer Linie dargestellt werden soll: A muß ungleich 0 sein Wenn der erste Punkt einer Linie nicht dargestellt werden soll: A muß 0 enthalten
	BC	–
	DE	–
	HL	–
Ausgabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Bemerkung		Diese Routine stellt ein, ob der erste Punkt einer zu zeichnenden Linie geplottet werden soll oder nicht.

Adresse:		**&BD4C**
Bezeichnung englisch		(GRA SET MASK)
Bezeichnung deutsch		Maske für das Zeichnen von Linien setzen.
Eingabeparameter	AF	enthält die Maske, mit der die Linien gezeichnet werden sollen
	BC	–
	DE	–
	HL	–

Ausgabeparameter	AF	–
	BC	–
	DE	–
	HL	–
Bemerkung		Die Maske, mit der die Linien gezeichnet werden sollen, ist als Binärzahl anzusehen, bei der die Bits gesetzt sind, die beim Zeichnen von Linien als gesetzte Punkte erscheinen sollen. Dabei entspricht der Wert &FF einer durchgezogenen Linie.

Adresse:		**&BD4F**
Bezeichnung englisch		(GRA CALCULATE COORDINATES)
Bezeichnung deutsch		Berechnet aus den gegebenen absoluten Grafik-Koordinaten bezüglich des Bildschirmmodus und der Lage des Ursprungs die physikalischen Koordinaten.
Eingabeparameter	AF	–
	BC	–
	DE	X-Koordinate bezüglich Ursprung
	HL	Y-Koordinate bezüglich Ursprung
Ausgabeparameter	AF	zerstört
	BC	–
	DE	physikalische X-Koordinate
	HL	physikalische Y-Koordinate
Bemerkung		Diese Routine rechnet die angegebenen Bildpunktkoordinaten auf die physikalischen Koordinaten um. Das bedeutet, daß die Koordinaten in DE und HL relativ zum Ursprung der Grafik mit 640*400 Bildpunkten angegeben werden und die Routine in DE und HL den physikalisch dadurch angesprochenen Pixel zurückliefert. Die physikalischen Koordinaten hängen vom jeweiligen Schreib-Modus ab. In MODE 0 hat man 160 (0 bis 159) *200 (0 bis 199) Pixels, in MODE 1 sind

320∗200 Pixels ansprechbar und in MODE 2 sogar 640∗200. Der Bildpunkt (0,0) liegt dabei in der linken unteren Ecke des Bildschirms. Die mit dieser Routine korrigierten Koordinaten können für eigene Routinen oder die einfachen Zeichen-Routinen des CPC verwendet werden.

Adresse:		**&BD52**
Bezeichnung englisch		(GRA FILL)
Bezeichnung deutsch		Füll-Routine zum Ausmalen von Flächen.
Eingabeparameter	AF	Farbstift, der – mit dem aktuellen Grafikfarbstift zusammen – die auszufüllende Fläche umschließt: zugleich wird die Fläche mit diesem Farbstift ausgefüllt
	BC	–
	DE	Größe des Puffers für die rekursive Flächenfüllfunktion
	HL	Adresse des Puffers für die rekursive FILL-Funktion
Ausgabeparameter	AF	zerstört
	BC	zerstört
	DE	zerstört
	HL	zerstört
Bemerkung		Diese Routine ist zum Ausfüllen der Fläche gedacht, in der sich der Grafik-Cursor zur Zeit des Aufrufs befindet. Die Routine füllt die ganze Fläche aus, überschreibt dabei alle Grafikpunkte, die nicht in der Grafikstiftfarbe bzw. in der angegebenen Farbe gesetzt sind. Trifft die Routine auf einen in einer der beiden Farben gesetzten Punkt, so beendet sie an dieser Stelle das Zeichnen und macht mit der Stelle weiter, an der die letzte Verzweigung an einem gesetzten Punkt war.

1.3 GRAFIKAUFBAU AUF DEM BILDSCHIRM

Grafikbildschirm und Textbildschirm sind bei allen CPC-Rechnern identisch. Textzeichen sind in diesem Falle also nichts anderes als Grafiksymbole, die innerhalb einer Matrix von 8 x 8 Bildschirmpunkten dargestellt werden. Dies hat den Vorteil, daß z. B. mit dem TAG-Befehl der Text beliebig innerhalb einer Grafik positioniert werden kann, wie wir in Kapitel 4 noch sehen werden. Doch um die Textdarstellung wollen wir uns hier nicht weiter kümmern.

Der Bildschirm läßt sich in 256 000 Bildschirmpunkte aufteilen. Diese Zahl ergibt sich, wenn man die Werte 640 und 400 miteinander multipliziert. Entsprechend der Größe des Bildschirms sind 640 Bildschirmpunkte (Pixels, Picture Elements) in einer Zeile vorhanden. Diese werden von 0 bis 639 durchgezählt. Analog werden die 400 Bildschirmpunkte in der Vertikalen von 0 bis 399 durchgezählt. In der Zählweise beginnt die Zeile am linken Bildschirmrand und die Spalte am unteren Bildschirmrand. Dies entspricht auch dem Koordinatensystem im mathematischen Sinn.

Nicht alle Bildschirmpunkte sind einzeln ansprechbar, wie man sich leicht errechnen kann, wenn man weiß, daß für den Bildschirmspeicher 16 KByte zur Verfügung stehen. Da für jeden Bildschirmpunkt ein Bit zuständig ist, müßten aber – um jeden Pixel einzeln darstellen zu können – 32 KByte Speicherplatz für den Bildschirm vorhanden sein. Diese Halbierung der ansprechbaren Bildpunkte erreicht man, indem man in Richtung der Y-Koordinate (Spalte) jeweils zwei Bildschirmpunkte einem Bit im RAM zuordnet. Die Auflösung in der Vertikalen beträgt also nur 200 Pixels. Sprechen Sie also die Y-Koordinaten 255 und 254 an, so erscheint jeweils der gleiche Bildschirmpunkt, da ja die Bildschirmpunkte trotzdem von 0 bis 399 durchgezählt werden. Testen können Sie dies, indem Sie

```
MOVE 200,255
DRAW 400,255
```

eingeben, womit Sie die erste Linie ziehen, und

```
MOVE 200,254
DRAW 400,254
```

womit Sie die zweite Linie ziehen, aber sich auf dem Bildschirm keine Änderung ergibt. Erst wenn Sie eingeben

```
MOVE 200,253
DRAW 400,253
```

wird die nächste Linie gezeichnet. Es genügt also, jeweils eine gerade oder ungerade Y-Koordinate anzugeben, was wir uns in Kapitel 2 für die schnellere Bearbeitung auch zunutze machen werden.

Statt 256 000 haben wir also nur 128 000 *ansprechbare* Bildschirmpunkte. Aber dies gilt auch nicht immer, da es vom verwendeten Bildschirmmodus (0, 1, 2) abhängig ist.

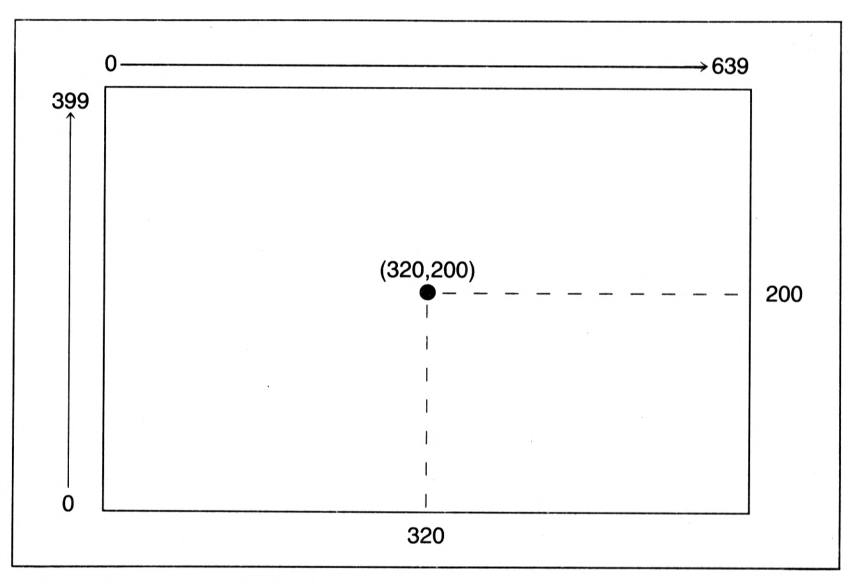


Abb. 1.1: Der Grafikbildschirm

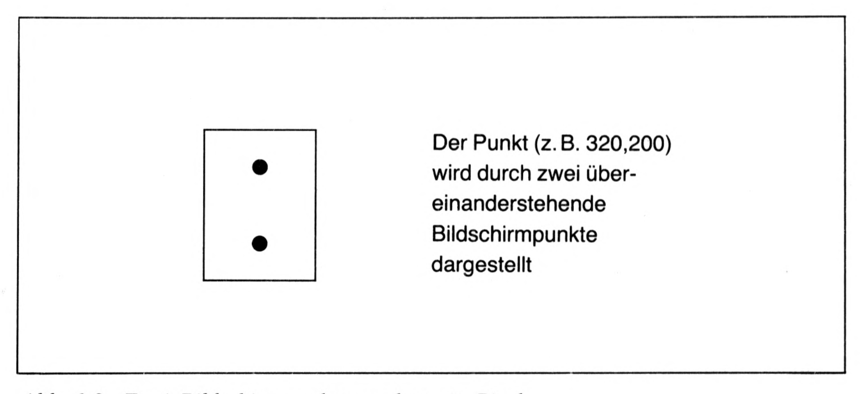


Abb. 1.2: Zwei Bildschirmpunkte ergeben ein Pixel.

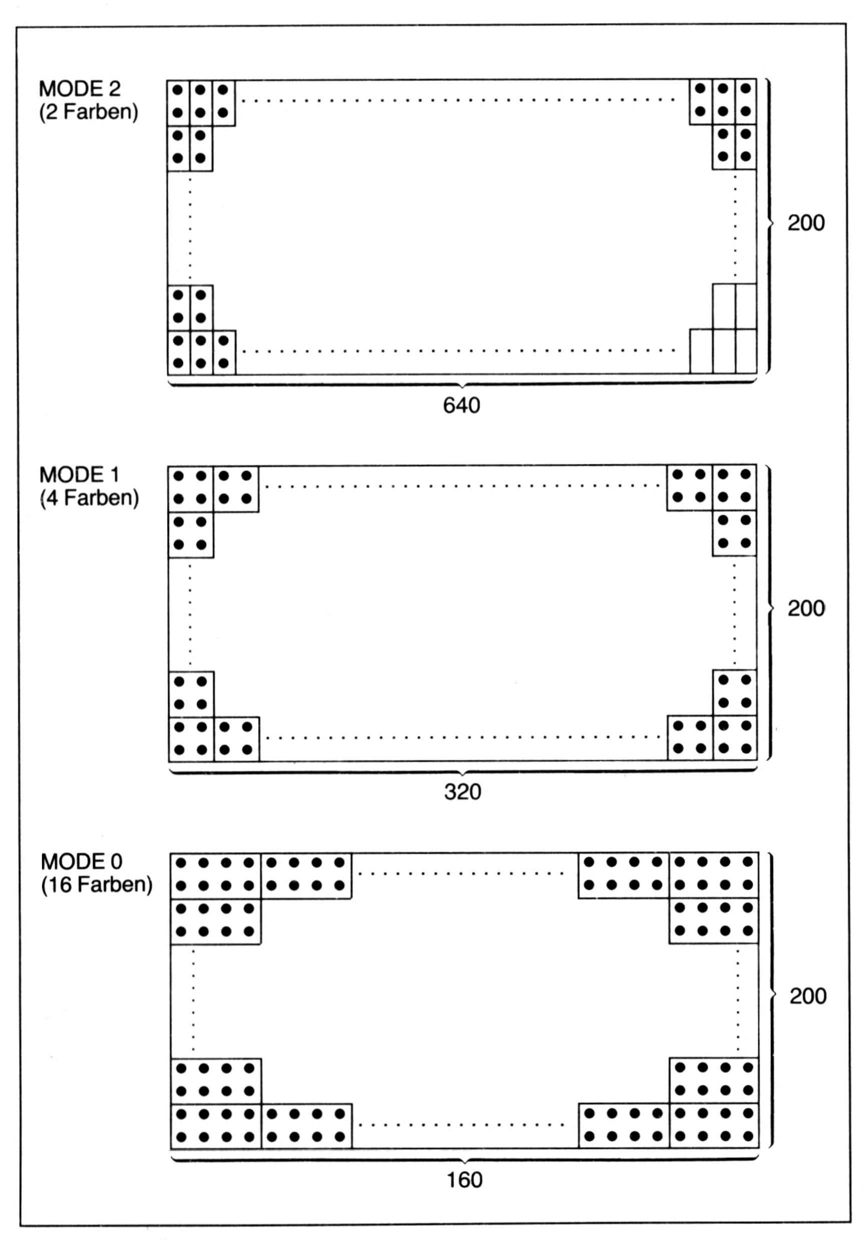


Abb. 1.3: Bildschirmpunkte in den verschiedenen Modi (jedes Kästchen steht für einen ansprechbaren Bildschirmpunkt)

Durch die Information in einem Bit kann man nur zwischen Farbe oder nicht Farbe unterscheiden. In diesem Fall hätte man nur die Wahlmöglichkeit zwischen Vorder- und Hintergrundfarbe. Dies ist natürlich nur im Bildschirmmodus 2 gegeben. Mehrere Farben lassen sich auf diese Weise also nicht darstellen. Wie geht nun der CPC vor, um doch mehrere Farben zu erzeugen?

Nimmt man zwei Bits, so lassen sich vier verschiedene Informationen damit codieren: 00, 01, 10, 11. Wenn man also die Auflösung um die Hälfte heruntersetzt und jeweils zwei Bits für einen ansprechbaren Bildschirmpunkt verwendet, so lassen sich vier verschiedene Farben (die Hintergrundfarbe und drei Vordergrundfarben) damit codieren. Zweckmäßigerweise werden nebeneinanderliegende Bildschirmpunkte verwendet, so daß wir diesmal in Richtung der X-Achse (Zeilen) die Hälfte der Auflösung verlieren. Uns stehen also vier Farben je Bildschirmpunkt, aber nur noch 320 ansprechbare Bildschirmpunkte je Zeile zur Verfügung. Die Gesamtzahl der ansprechbaren Bildschirmpunkte ist also auf 64 000 gesunken.

Vier Farben haben dem Bildschirmmodus 1 entsprochen. Im Modus 0 stehen uns 16 Farben zur Verfügung, die wir in vier Bits codieren können. Unsere Auflösung in der Horizontalen wird also nochmals halbiert, so daß uns pro Bildschirmzeile nur 160 ansprechbare Punkte zur Verfügung stehen (insgesamt 32 000), dafür aber 16 Farben. Abb. 1.3 veranschaulicht die Zusammenhänge, wobei innerhalb eines Kastens jeweils ein ansprechbarer Bildschirmpunkt dargestellt ist.

Der Rahmen des Bildschirms ist von dieser Darstellung unabhängig und kann mittels des BORDER-Befehls frei gewählt werden.

Die verschiedenen möglichen Farben (1, 4, 16) werden durch den INK-Befehl einer Nummer zwischen 0 und 15 zugeordnet, da die Farbnummern von 0 bis 26 sich natürlich nicht alle in den entsprechenden Bitmustern darstellen lassen. Dabei können einer Farbauswahlnummer (0 bis 15) auch zwei Farben zugeordnet werden, die dann abwechselnd am Bildschirm blinken. Die Blinkdauer kann mit dem Befehl SPEED INK vom Anwender selbst festgelegt werden.

1.4 VORHANDENE GRAFIKBEFEHLE

Obwohl der Schneider CPC ein sehr umfangreiches BASIC aufweist, sind die Grafikbefehle etwas zu kurz gekommen. Trotzdem sind mit ihnen sicherlich alle nur denkbaren grafischen Figuren darstellbar. Besondere geometrische Figuren wie Rechteck, Block, Quader, Kreis, Ellipse oder einen Stern haben wir in Kapitel 2 sowohl als Unterprogramm in BASIC als auch als Maschinenprogramm mit entsprechendem RSX-Befehl dargestellt.

Da das Handbuch die Grafikbefehle ausreichend beschreibt und Sie im Verlauf dieses Buches viele Beispiele für ihre Anwendung finden, wollen wir uns hier nur kurz auf die wesentlichen Merkmale der Grafikbefehle beschränken.

Als erstes wollen wir auf die Positionierung des Grafik-Cursors eingehen, da sie von der Vorgehensweise bei den meisten Rechnern abweicht. Beim CPC befindet sich der Grafik-Cursor immer an der Position, die durch einen Grafikbefehl zuletzt angesprochen wurde. Aus diesem Grunde ist es beim Ziehen einer Linie z. B. auch unnötig, den Anfangs- und Endpunkt der Linie anzugeben, da der Rechner davon ausgeht, daß der Anfangspunkt der Linie an der aktuellen Position des Grafik-Cursors liegt. Daher braucht nur der Endpunkt der Linie angegeben zu werden, mit dem Resultat, daß sich nach Ziehen der Linie der Grafik-Cursor am Ende der Linie befindet. Dieser Umstand wird in Kapitel 4 (Liniendiagramme) in besonderer Weise ausgenutzt.

Das Positionieren des Grafik-Cursors kann mit den Befehlen MOVE, MOVER, PLOT, PLOTR, TEST sowie TESTR erfolgen. Der Unterschied zwischen MOVE und PLOT macht sich nur am Bildschirm bemerkbar: Bei PLOT wird der Punkt auch gezeichnet. Der eigentliche Befehl zum Positionieren des Cursors lautet MOVE bzw. MOVER. Die Befehle TEST bzw. TESTR werden als Funktion verwendet, d. h. sie geben einen Wert (die Farbnummer an dem angegebenen Punkt) zurück.

Eine weitere besondere Eigenschaft der CPC-Rechner ist die zweifache Auslegung der Befehle als „Befehl" und „BefehlR". Ohne die Verwendung des R bedeuten die im Befehl angegebenen Parameter eine *absolute* Positionierung, d. h. der angesprochene Bildschirmpunkt ist in Koordinaten bezüglich des Anwender-Koordinatenursprungs (siehe unten) angegeben. Bei Verwendung eines Befehls mit dem R wird der Punkt *relativ* zur aktuellen Cursor-Position ausgegeben. Befindet sich der Grafik-Cursor z. B. in der Position (0,0) und zeichnen Sie mit DRAW eine Linie zur Position (100,100), so befindet sich der Grafik-Cursor anschließend an der *absoluten* Position (100,100), und die Linie ist zwischen den absoluten Werten (0,0) und (100,100) gezogen. Geben Sie anschließend sofort den Befehl DRAWR mit den Parametern (100,100) ein, so wird die Linie bis zum absoluten Punkt (200,200) gezogen.

Weiter bemerkenswert ist der ORIGIN-Befehl, der es erlaubt, den Ursprung des Anwender-Koordinatensystems (absoluter Punkt [0,0]) festzulegen. Wollen Sie z. B. ein karthesisches Koordinatensystem auf dem Bildschirm mit allen vier Quadranten darstellen, so genügt der Befehl ORIGIN 320,200, um den Ursprung des Anwender-Koordinatensystems auf dem Bildschirmmittelpunkt zu fixieren. Dies erspart z. B. das Umrechnen bei Funktionswerten, wenn Sie in diesem Koordinatensystem eine Funktion darstellen möchten.

Die zweite Anwendung des ORIGIN-Befehls ist die Definition des – einzigen – Grafikfensters. Zur Definition des Grafikfensters sei hier auf die entsprechenden Betriebssystem-Routinen hingewiesen. Die vorgegebenen Koordinaten müssen nicht mit den tatsächlichen Koordinaten des Grafikfensters übereinstimmen.

Sollten Sie einmal nicht über die genaue Position des Grafik-Cursors informiert sein, so helfen Ihnen die Befehle XPOS und YPOS weiter, die Ihnen die augenblicklichen Koordinaten des Grafik-Cursors zurückgeben. Diese beiden Befehle – besser: Funktionen – lassen sich auch innerhalb eines Programms verwenden.

Durch die Befehle ORIGIN, XPOS und YPOS sind also dem Anwender Hilfsmittel gegeben, die nichts direkt mit dem Zeichnen von Grafiken zu tun haben, aber in manchen Anwendungsfällen von Bedeutung sind. Wünschenswert wäre allerdings noch die Darstellung verschiedener geometrischer Figuren mit einem einzigen Befehl gewesen. Damit Ihnen solche Befehlserweiterungen zur Verfügung stehen, bringen wir die wichtigsten in Kapitel 2.

Die bisher besprochenen Befehle gelten sowohl für den CPC 464 als auch den CPC 664 und CPC 6128. Die neueren Rechner 664 und 6128 weisen gegenüber dem Basismodell 464 im Bereich der Grafikbefehle einige Besonderheiten auf, auf die wir an dieser Stelle eingehen wollen.

Zunächst ist zu erwähnen, daß bei den Befehlen DRAW, DRAWR, MOVE, MOVER, PEN, PNOT und PLOTR der Farbstift-Modus mit angegeben werden kann. Dieser Farbstift-Modus gibt an, ob die angegebenen Punkte oder Linien direkt auf den Bildschirm geschrieben oder die neuen Daten mit bereits am Bildschirm befindlichen Punkten verknüpft werden sollen. Als Verknüpfung stehen AND, OR und XOR zur Verfügung. In Kapitel 5 werden die Möglichkeiten des Farbstift-Modus weiter ausgeführt.

Weiterhin verfügen der 664 und 6128 über einen FILL-Befehl, mit dem abgeschlossene Gebiete auf dem Bildschirm sehr einfach eingefärbt werden können. Zu beachten ist dabei, daß der abgeschlossene Bildschirmbereich unbedingt von der im FILL-Befehl angegebenen Farbe oder der aktuellen Farbe des Grafikpens umschlossen sein muß, um zu verhindern, daß der ganze Bildschirm mit dieser Farbe gefüllt wird.

Weniger als Grafikbefehl, aber auch in diesem Bereich nützlich, kann der COPYCHR$-Befehl angesehen werden. Mit ihm können Zeichen von einer Stelle des Bildschirms auf eine andere umkopiert werden.

Bei den neuen CPC-Rechnern ist es weiterhin möglich, den Hintergrund und die Stiftfarbe – wie beim Text auch – durch BASIC-Befehle festzulegen.

Hierzu werden die Befehle GRAPHICS PAPER und GRAPHICS PEN analog den Befehlen PEN und PAPER herangezogen.

Wer beim CPC 464 das Beschreiben des Bildschirms mit dem Strahlrücklauf synchronisieren wollte, war auf eine eingebaute Routine (Adresse: &BD19) angewiesen. Beim CPC 664 und 6128 kann man den Flackereffekt mit dem FRAME-Befehl unterdrücken.

Als letzte Neuerung möchten wir den MASK-Befehl erwähnen. Besonders bei Diagrammen oder technischen Zeichnungen ist es sehr oft sinnvoll, gestrichelte Linien einzusetzen. Der MASK-Befehl ermöglicht Ihnen, aufgrund einer einzugebenden Maske (1 Byte) die Strichart Ihrer Linie selbst festzulegen.

1.5 DER ZEICHENSATZ UND SEINE GRAFIKELEMENTE

Neben der normalen hochauflösenden Grafik, die sich mit den im Abschnitt „Vorhandene Grafikbefehle" besprochenen Befehlen realisieren läßt, weist der CPC noch einen umfangreichen Zeichensatz mit hohem Anteil von Grafikzeichen aus. Mit ihm wollen wir uns in diesem Abschnitt beschäftigen.

Allgemeines zu den Grafikelementen des Zeichensatzes

Neben dem normalen Alphabet in Groß-/Kleinschrift, den üblichen Sonderzeichen und den Ziffern, die sich in den Bereichen (dezimal) 32–90 und 97–125 des Zeichensatzes befinden, sowie einigen speziellen Sonderzeichen (engl. Pfundzeichen, Copyrightzeichen, Indizes, gespiegelten Sonderzeichen und einem griechischen Alphabet) in dem Bereich 163 bis 191 sind sehr viele grafische Sonderzeichen vorhanden, die ausführlich im Handbuch aufgelistet sind. Der besseren Übersichtlichkeit wegen haben wir die einzelnen Bereiche für Sie im folgenden kurz zusammengefaßt:

126–144 Karomuster in verschiedenen Ausprägungen
145–159 Strichmuster mit geraden Linien
192–205 Strichmuster mit Diagonallinien
206, 207
und
216–223 Karierte Muster
208–211 Randlinien
212–215 Dreiecke
224, 225 Glückliches, trauriges Gesicht
226–229 Kartensymbole
230–239 Verschiedene Symbole wie leerer und voller Kreis, leeres und

volles Quadrat, männliches und weibliches Symbol, Achtel- und Viertelnote, Stern und Rakete
240–243 Pfeile in verschiedenen Richtungen
244–247 Dreiecke mit Spitze in verschiedenen Richtungen
248–251 Tanzendes Männchen
252, 255 Fallende Bombe und Wolke
254, 255 Doppelpfeile nach oben/unten und rechts/links

Man sieht also, daß gegenüber den Zeichensätzen in anderen Rechnern auch besondere Feinheiten berücksichtigt wurden, wie z. B. Noten und das tanzende Männchen. Auch Kartenspieler kommen auf ihre Kosten, da sie als Anwender diese Symbole nicht mehr selbst definieren müssen. Der griechische Teil des Zeichensatzes dürfte besonders für Mathematiker und Physiker von Interesse sein.

Die Blockgrafik-Symbole im Bereich von 128 (Leerzeichen) bis 143 (inverses Leerzeichen) können dazu genutzt werden, auch geometrische Figuren ohne Verwendung der hochauflösenden Grafik darzustellen. Besonders die Liniensymbole im Bereich 144 bis 159 können sehr gut als Rahmen bei Anwenderprogrammen (z. B. Menüs oder Anwenderhinweisen) verwendet werden, ohne daß man sich für die Darstellung auf Bitebene begeben muß. Die Verwendung der Linien und Diagonalen wird einem Modellbahner sicherlich auch die Erstellung eines Gleisbildstellpultes vereinfachen. Der Anwendung der vorhandenen Grafikzeichen sind nur durch Ihre Phantasie Grenzen gesetzt.

Weitere Sonderzeichen sind auch im Bereich 0 bis 31 des Zeichensatzes vorhanden. Hier ist vor dem Aufruf des Zeichens zunächst ein CHR$(1) anzugeben und dann innerhalb eines CHR$()-Befehls die entsprechende Nummer.

Hier die Liste der Sonderzeichen:

0: Quadrat
1: linker und oberer Strich
2: kopfstehendes T
3: rechter und unterer Strich
4: Blitz/Hochspannung
5: Quadrat mit X in der Mitte
6: Abhakzeichen
7: Glockenzeichen (entspricht dem „Bell“-Symbol vom Fernschreiber). Der Steuercode 7 ist ein kurzer Piepston
8: spitzer Pfeil nach links
9: spitzer Pfeil nach rechts
10: spitzer Pfeil nach unten
11: spitzer Pfeil nach oben

12: spitzer Doppelpfeil nach unten
13: abgewinkelter Pfeil nach links (entsprechend Carriage Return/Wagenrücklauf)
14: Kreis mit X in der Mitte
15: Kreis mit Kringel in der Mitte
16: Quadrat mit Querstrich
17: Vollkreis mit rechts offenem U
18: Vollkreis mit kopfstehendem, rechts offenem U
19: Vollkreis mit kopfstehendem, links offenem U
20: Vollkreis mit links offenem U
21: X mit Haken unten links/durchgestrichenes Abhakzeichen
22: Symbol für Rechteckschwingung
23: 90 Grad nach rechts gedrehtes T
24: X mit Strich oben und unten
25: senkrechter Strich mit Punkt in der Mitte
26: an der Längsachse gespiegeltes Fragezeichen
27: Kreis mit Querstrich waagerecht
28: Vollquadrat mit links offenem U
29: Vollquadrat mit kopfstehendem links offenem U
30: Vollquadrat mit kopfstehendem rechts offenem U
31: Vollquadrat mit rechts offenem U

Auch in dieser Liste finden sich sicherlich einige Sonderzeichen, die im einen oder anderen Anwendungsfall sehr nützlich sein können.

Anwendungsbeispiel: Gleisbildstellpult

Normalerweise können die (nicht mit der Tastatur erreichbaren) Grafikzeichen des Zeichensatzes nur mit dem CHR$()-Befehl verwendet werden. Dazu ist jeweils die Nummer innerhalb des Zeichensatzes als Argument des CHR$()-Befehls einzutragen. Besonders bei häufiger Verwendung von Grafikzeichen kann dies in eine langwierige Sucharbeit im Handbuch ausarten, und auch das Programm ist nicht sehr übersichtlich, da die Rückinterpretation von Nummer des Zeichensatzes in Sonderzeichen ebenfalls sehr zeitraubend ist.

Man kann natürlich auch zur schnelleren und übersichtlicheren Programmierung einzelnen Tasten, insbesondere der Zehnertastatur, verschiedene Grafiksymbole zuordnen. Im folgenden wollen wir jedoch ein anderes Verfahren vorstellen.

Als Demonstrationsbeispiel wollen wir den Plan eines kleinen Gleisbildstellpultes für eine Modelleisenbahn heranziehen. Dazu eignen sich besonders die

Sonderzeichen der Liniengrafik, da hier Kreuzungen, die Diagonalen und Weichen darstellbar sind. Wem die doch recht eckigen Weichen etwas zu unübersichtlich sind, der kann sich diese Zeichen auch noch sehr einfach selbst umdefinieren.

Das Listing zeigt Programm 1.1.

Neben der Definition der Bildschirmfenster gliedert sich das Listing in zwei Teile. Ab Zeile 2000 werden die verwendeten Grafikzeichen im Bildschirmfenster #2 am unteren Bildschirmrand dargestellt, und ab Zeile 3000 befindet sich die eigentliche Ausgabe des Gleisbildstellpultes.

```
1000 REM ------------------------------
1010 REM --- Grafikzeichen fuer     ---
1020 REM ---    Gleisbildstellpult  ---
1030 REM ------------------------------
1040 :
1041 MODE 1
1042 :
1050 WINDOW #0,1,40,1,22
1060 WINDOW #2,1,40,23,25
1070 :
2000 REM ------------------------------
2010 REM --- Hilfsprogramm: Grafik- ---
2020 REM --- zeichen auf den Bild-  ---
2030 REM --- schirm                 ---
2040 REM ------------------------------
2050 :
2060 FOR i=145 TO 159
2070    PRINT#2,CHR$(i);" ";
2080 NEXT
2090 :
2100 FOR i=192 TO 205
2110    PRINT#2,CHR$(i);" ";
2120 NEXT
2130 :
3000 REM ------------------------------
3010 REM --- Zeichnen des Gleis-    ---
3020 REM ---      stellpultes       ---
3030 REM ------------------------------
3040 :
3050 CLS #0
3060 PRINT #0,"BC"
3070 PRINT #0,"BC"
3080 PRINT #0,"       A§           "
3090 PRINT #0,"        A§            "
3100 PRINT #0,"         A§             "
3110 PRINT #0,"                                "
3120 PRINT #0,"                                "
3130 PRINT #0,"                                "
3140 PRINT #0,"                                "
3150 PRINT #0,"A§"
3160 PRINT #0,"A§"
```

Programm 1.1: Gleisbildstellpult

Zunächst jedoch noch etwas zu den Fenstern. Bei der Programmeditierung wird der Bildschirm automatisch von oben nach unten beschrieben. Wenn wir unsere Sonderzeichen am unteren Bildschirmrand darstellen, werden diese dann bei der Programmeditierung überschrieben. Daher haben wir zwei Fenster definiert, wobei das Fenser #2 die unteren drei Bildschirmzeilen beinhaltet, in dem wir unsere Grafikzeichen darstellen wollen. Das normale Bildschirmfenster (#0) wird um diese drei Zeilen verkürzt. Außerdem wird „imaginär“ noch das Bildschirmfenster #1 verwendet, wenn auch nicht innerhalb des Programms, sondern bei der Programmeditierung.

Ab Zeile 2000 werden die benötigten Grafikzeichen des Zeichensatzes in Bildschirmfenster #2 ausgegeben. Dies sind die Zeichen mit den Nummern 145 bis 159 und 192 bis 205 (Strichmuster mit Geraden und Linien und Diagonallinien).

Bis 2130 sind alle Zeilen vor der Editierung der eigentlichen Zeichnung einzugeben. Außerdem ist noch eine Funktionstaste mit

```
WINDOW SWAP 0,1
```

zu belegen, so daß wir unsere Arbeit entweder in Bildschirmfenster #0 oder #1 durchführen können. Bei einem Austausch der Fenster wird der Bildschirm nicht gelöscht, was wir uns zunutze machen.

Wenn Sie nun das Programm gestartet haben, werden Sie feststellen, daß Sie bei der Programmeditierung die untersten drei Zeilen mit den Grafikzeichen nicht erreichen können. Löschen Sie jetzt zunächst Bildschirmfenster #0, und geben Sie anschließend mit Hilfe der belegten Funktionstaste den Befehl WINDOW SWAP 0,1 ein. Nun können Sie im oberen Bildschirmbereich ein Programm editieren und – sobald Sie ein Grafikzeichen benötigen – dies mit dem COPY-Cursor an die entsprechende Programmstelle kopieren. Sobald Sie den unteren Bereich erreichen, ist durch den Befehl WINDOW SWAP 0,1 sicherzustellen, daß Sie Grafikzeichen nicht überschreiben. Zweckmäßigerweise sollte dann das Fenster #0 wieder gelöscht werden. Aber auch, wenn Sie den Grafikbereich überschrieben haben, können Sie die Sonderzeichen durch RUN wieder am unteren Teil des Bildschirms darstellen.

Zur Vereinfachung sollten Sie auch ähnliche Ausgaben aus bereits programmierten Programmzeilen probieren.

Das Ergebnis unseres Programms ist in Abb. 1.4 dargestellt.

In unserem Listing werden natürlich die Grafikzeichen bei einem normalen Drucker nicht ausgegeben. Hier ist es sehr hilfreich, eine Hardcopy-Routine zu haben, da man dann das Programm abschnittsweise auf dem Bildschirm

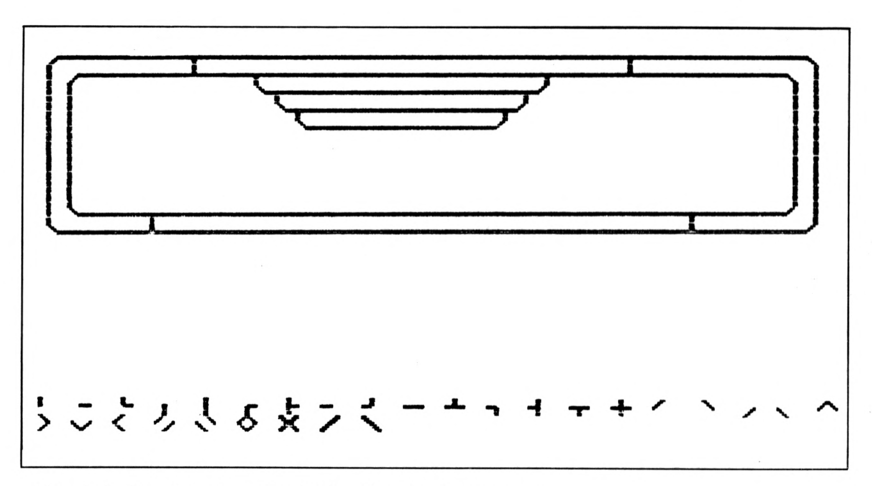


Abb. 1.4: Ergebnis des Gleisbildstellpunkt-Programms

darstellt und mittels der Hardcopy-Routine ausgibt, wie wir es in Abb. 1.5 durchgeführt haben.

Mit diesem Trick vermeiden Sie also bei aufwendiger Programmierung von Grafiksonderzeichen die unübersichtlichen CHR$()-Befehle und eine langwierige Suche sowohl beim Erstellen des Programms als auch beim späteren Ändern. Natürlich lassen sich nicht nur Gleisbildstellpulte in dieser Art darstellen. Ein wenig Übung ist jedoch schon gefragt, wie Sie bei einem Versuch leicht feststellen können.

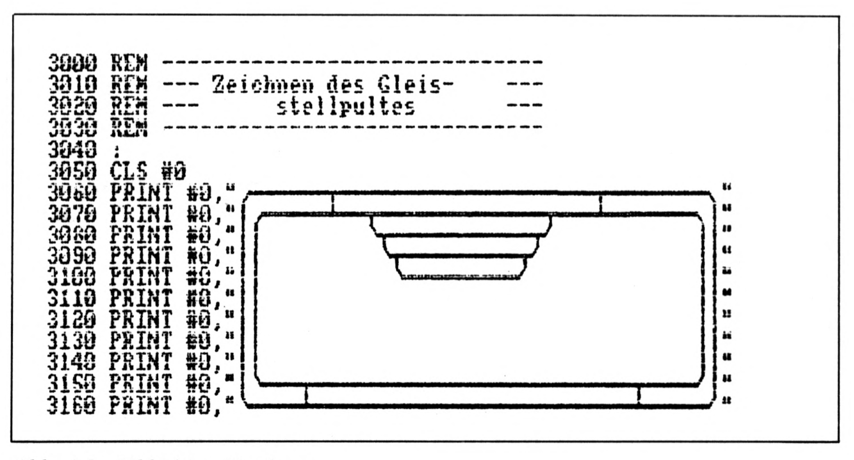


Abb. 1.5: Bildschirm-Hardcopy

Kapitel 2
Erweiterung der Grafikbefehle mit Anwendungsbeispielen

Obwohl Schneider-Rechner einen umfangreichen BASIC-Befehlssatz haben und auch über die Grafikbefehle PLOT, PLOTR, DRAW, DRAWR, MOVE, MOVER, ORIGIN und TEST sowie MASK, FILL und FRAME beim 664 und 6128 verfügen, bleiben bei der Erstellung von komplizierteren Grafiken immer noch einige Probleme ungelöst.

In diesem Kapitel wollen wir die Thematik „Grafikbefehle“ aufgreifen und einige Ergänzungen zu den Grafikbefehlen produzieren, wobei wir nacheinander einen Rechteck-Befehl, Block-Befehl, Quader-Befehl, Kreis-Befehl, Radius-Befehl und Stern-Befehl besprechen wollen. Für diejenigen, die der Maschinensprache-Programmierung nicht mächtig sind, aber auch zur Einführung in die Thematik, werden zunächst alle Befehle als BASIC-Unterprogramme dargestellt.

2.1 GRAFIKERWEITERUNGEN ALS BASIC-UNTERPROGRAMME

Anders als z. B. beim Commodore 64 ist das BASIC des CPC 464 bereits mit einer Anzahl von Grafikbefehlen ausgestattet, auf die wir uns im folgenden stützen wollen.

In der Hauptsache werden wir dabei die Befehle MOVE und MOVER (Setzen des Grafik-Cursors), PLOT und PLOTR (Ausgabe eines einzelnen Punktes) sowie DRAW und DRAWR (Zeichnen einer Linie) verwenden.

Die Variablen bei den Unterprogrammen wurden sinngemäß verwendet, und wir weisen an dieser Stelle ausdrücklich darauf hin, daß bei Verwendung der nachfolgend beschriebenen BASIC-Unterprogramme diese Variablen eine

Verwendung ausschließlich als Parameter finden sollten. Die entsprechenden Koordinaten zur Ausgabe einer geometrischen Figur werden dem Unterprogramm über diese Parameter mitgeteilt, so daß die Unterprogramme unabhängig von der jeweiligen Verwendung werden. Näheres dazu finden Sie in den weiteren Kapiteln.

Im folgenden sind die Unterprogramme so angeordnet, daß man alle Unterprogramme und Testdaten des vorliegenden Kapitels in einem Programm zusammenfassen kann.

Rechteck

Am einfachsten ist ein Unterprogramm zur Ausgabe eines Rechtecks zu realisieren. Da – wie soeben erwähnt – die Position und Größe des Rechtecks dem Unterprogramm mitgeteilt werden soll, benötigen wir einige Variablen, die als Eingabeparameter für das Unterprogramm fungieren. Hier folgt zunächst das Listing für das Unterprogramm zum Zeichnen eines Rechtecks (Programm 2.1).

Zur Veranschaulichung der Parameter wollen wir Abb. 2.1 betrachten.

Ein Rechteck ist also durch die vier Koordinaten obenx, obeny, untenx und unteny eindeutig definiert. Die beiden nicht explizit bezeichneten Ecken des

```
50000 REM *****************************
50010 REM *****************************
50020 REM ***                       ***
50030 REM ***   Grafische           ***
50040 REM ***                       ***
50050 REM ***      Unterprogramme   ***
50060 REM ***                       ***
50070 REM *****************************
50080 REM *****************************
50090 :
50100 :
50110 :
50120 REM -----------------------------
50130 REM ---       Rechteck        ---
50140 REM -----------------------------
50150 :
50160    MOVE obenx,obeny
50170    DRAW obenx,unteny,farbe
50180    DRAW untenx,unteny,farbe
50190    DRAW untenx,obeny,farbe
50200    DRAW obenx,obeny,farbe
50210 :
50220 RETURN
50230 :
```

Programm 2.1: Rechteck-Unterprogramm

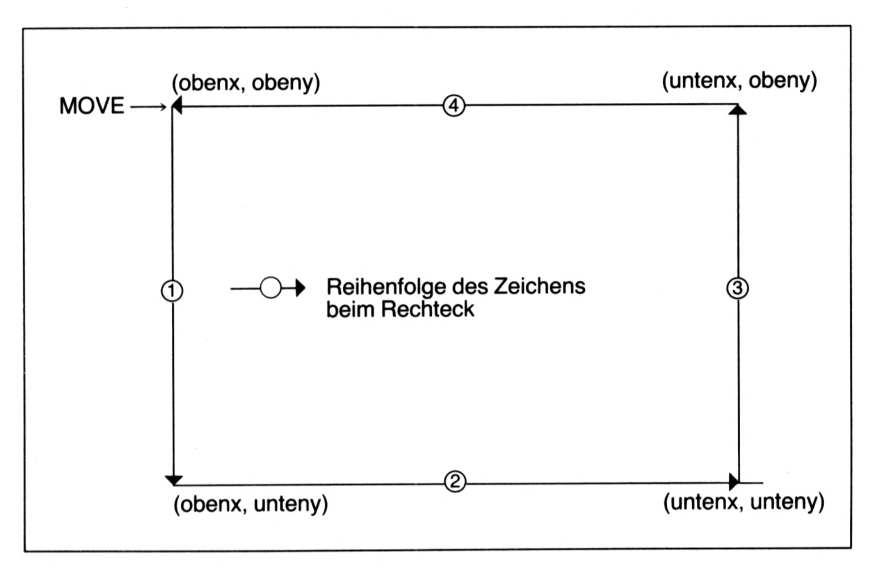


Abb. 2.1: Funktionsschema zum Rechteck-Befehl

Rechtecks ergeben sich aus den beiden diagonal gegenüberliegenden – bestimmten – Ecken. Eine andere Möglichkeit zur Darstellung des Rechtecks (Angabe von Länge und Breite) lernen wir später noch kennen.

Beim Zeichnen des Rechtecks kommt uns zugute, daß der Grafik-Cursor jeweils an der letzten gezeichneten Position verbleibt, bis ein neuer Befehl erfolgt. Aus diesem Grunde reicht es, jeweils die Zielkoordinaten der gewünschten Linie anzugeben.

Über den Parameter farbe kann noch eine entsprechende Farbe aus der INK-Palette ausgewählt werden. Auch diese Variable ist Eingabeparameter und kann für jeden Unterprogrammaufruf neu vom Anwender bestimmt werden.

Testen wir das Unterprogramm mit Hilfe von Programm 2.2.

Zuerst sehen wir einen Vorspann in den Zeilen 1000 bis 1080, in dem einige Farbwerte umgesetzt, die PEN- und PAPER-Zuordnung festgelegt und der Bildschirm gelöscht wird. Dieser Vorspann dient zur Sicherheit, wenn Sie im vorher gelaufenen Programm diese Parameter verstellt haben. Am sichersten ist das Zurücksetzen des Rechners (mittels SHIFT/CTRL/ESC durchzuführen).

Nachdem zunächst die Farbe festgelegt wurde, werden in einer Schleife von allen Ecken aus immer größer werdende Rechtecke gezeichnet, die sich größ-

```
1000 REM ------------------------------
1010 REM --- Vorspann und Testdaten ---
1020 REM ------------------------------
1030 :
1040 INK 0,10 : PAPER 0
1050 INK 1,6 : PEN 1
1060 INK 2,21
1070 CLS
1080 :
1090 REM ------------------------------
1100 REM -- Testdaten fuer Rechteck ---
1110 REM ------------------------------
1120 :
1130 farbe = 1
1135 GOTO 5060
1140 :
1150 FOR i=12 TO 164 STEP 5
1160 :
1170    obenx  = i
1180    obeny  = i
1190    untenx = 2*i
1200    unteny = 2*i
1210    GOSUB 50120
1220 :
1230    obenx  = 640-2*i
1240    untenx = 640-i
1250    GOSUB 50120
1260 :
1270    obeny  = 400-i
1280    unteny = 400-2*i
1290    GOSUB 50120
1300 :
1310    obenx  = i
1320    untenx = 2*i
1330    GOSUB 50120
1340 :
1350 NEXT
1360 :
1370 GOSUB 40000
1380 :
```

Programm 2.2: Anwendung des Rechteck-Unterprogramms

tenteils überlagern. Das Ergebnis der abstrakten Zahlen in ihrer grafischen Darstellung ist in Abb. 2.2 dargestellt.

In diesem wie auch in allen anderen Fällen ist es anzuraten, jeweils die Parameter in den Beispielprogrammen etwas abzuändern, so daß Sie feststellen können, welche Wirkung die einzelnen Programmzeilen haben.

Die Reihenfolge der Ausgabe ist folgende: Als erstes wird der „Strang" von unten links zur Mitte ausgegeben, wobei von der Punktverteilung her Quadrate vorliegen, deren Kantenlänge doppelt so groß wie der Abstand von den beiden Hauptachsen ist. Für den zweiten „Strang" werden die Y-Koordinaten

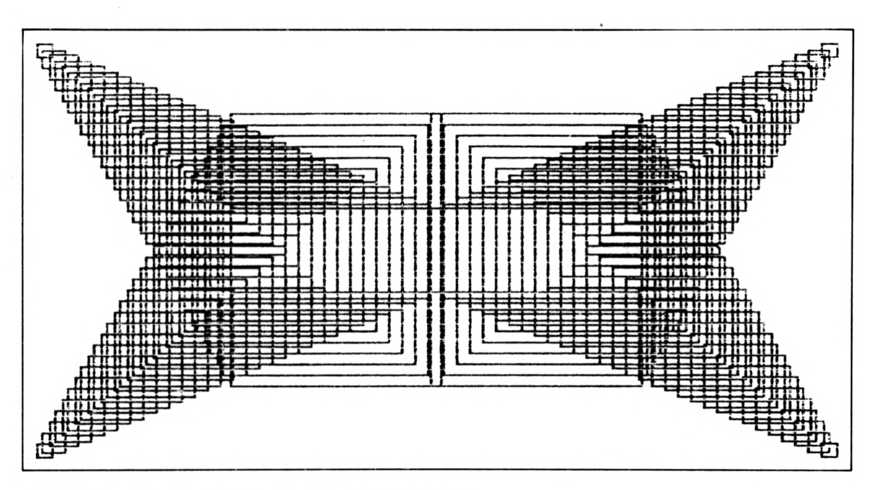


Abb. 2.2: Hardcopy nach dem Programmlauf

beibehalten und lediglich die X-Koordinaten dahingehend geändert, daß von der rechten Seite aus gezeichnet wird. Beim dritten „Strang“ werden die Y-Koordinaten geändert, so daß in diesem Fall die Rechtecke von oben rechts weggezeichnet werden. Für die letzte Ecke reicht es, wieder die X-Koordinaten alleine zu ändern.

Wie Sie vielleicht festgestellt haben, ist es nicht notwendig, daß analog zu Abb. 2.1 die Koordinate untenx größer ist als die Koordinate obenx. D. h. die Zuordnung der Wertepaare zu den Variablennamen kann auch vertauscht werden, was zwar aus Dokumentationsgründen nicht zu empfehlen ist, manchmal im Programm jedoch zu erheblichen Vereinfachungen führt.

In Zeile 1370 wird ein Unterprogramm ab Zeile 40000 aufgerufen, in dem neben der Benutzermeldung „Bitte Taste druecken“ lediglich der Tastendruck abgewartet wird. Falls dieser Tastendruck ein h ist, so wird ein weiteres Unterprogramm aufgerufen, mit dem das größtmögliche Rechteck ausgegeben wird (Umrahmung des Bildschirms) und anschließend die Hardcopy-Routine. Die Grafikbefehle zur Einrahmung des Bildschirms wurden bewußt mit absoluten Zahlen versehen, um Ihnen auch den Einsatz in anderen Programmen (ohne das Unterprogramm „Rechteck“) zu ermöglichen.

Die beiden Unterprogramme sehen Sie in Programm 2.3.

Zeile 40060 wurde eingefügt, um für die Hardcopy-Ausdrucke die Benutzermeldung „Bitte Taste druecken“ zu unterbinden.

Wir wollen nun das Rechteck zum Block erweitern, d. h. wir beschreiben nun die Vorgehensweise zum Ausfüllen des Rechtecks.

```
40000 REM ----------------------------
40010 REM --- Warteschleife auf     ---
40020 REM --- Tastendruck und       ---
40030 REM --- Loeschen des Bild-    ---
40040 REM --- schirmes              ---
40050 REM ----------------------------
40060 GOTO 40110
40070 :
40080 LOCATE 5,24
40090 PRINT"Bitte Taste druecken"
40100 :
40110    a$=INKEY$
40120    IF a$="" THEN 40110
40130 :
40140    IF a$="h" THEN GOSUB 41000
40150 :
40160 :
40170 CLS
40180 RETURN
40190 :
41000 REM ----------------------------
41010 REM --- Hardcopy mit Rahmen   ---
41020 REM ----------------------------
41030 :
41040    MOVE 0,0
41050    DRAW 639,0,1
41060    DRAW 639,399,1
41070    DRAW 0,399,1
41080    DRAW 0,0,1
41090 :
41100    CALL &A080
41110 :
41120 RETURN
41130 :
```

Programm 2.3: Die Unterprogramme zum Warten auf einen Tastendruck und für die Hardcopy

Block

Auch bei der Ausgabe eines Blocks benutzen wir die Grafikbefehle MOVE und DRAW. Hier setzen wir jedoch viele Linien nebeneinander, so daß im

```
51000 REM ----------------------------
51010 REM ---       Block           ---
51020 REM ----------------------------
51030 :
51040    FOR block=obenx TO untenx
51050       MOVE block,obeny
51060       DRAW block,unteny,farbe
51070    NEXT
51080 :
51090 RETURN
51100 :
```

Programm 2.4: Das Block-Unterprogramm

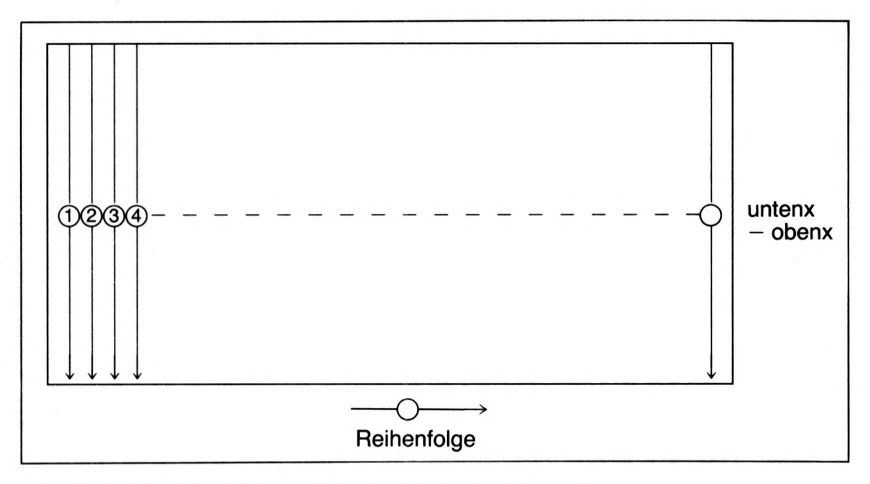


Abb. 2.3: Die Vorgehensweise beim Zeichnen eines Blocks

Endeffekt ein ausgefülltes Rechteck entsteht. Das Unterprogramm zeigt Programm 2.4.

Die Vorgehensweise wird in Abb. 2.3 deutlich.

Die Variablen wurden bewußt identisch mit denen zum Ausgeben eines Rechtecks gewählt, um dem Anwender in seinem Programm eine einfache Möglichkeit zu bieten, zwischen beiden Darstellungsarten zu wählen. Mit den

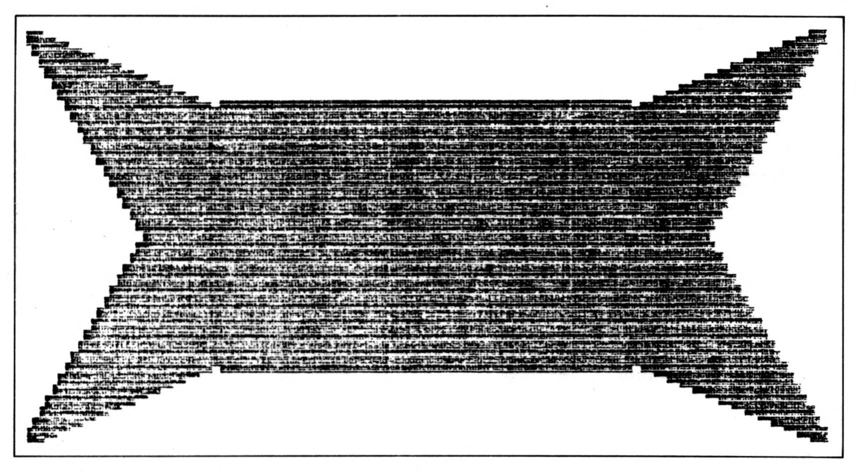


Abb. 2.4: Die Ausgabe von Programm 2.5

```
1390 REM ------------------------------
1400 REM ---  Testdaten fuer Block  ---
1410 REM ------------------------------
1420 :
1430 FOR i=12 TO 164 STEP 5
1440    obenx  = i
1450    obeny  = i
1460    untenx = 2*i
1470    unteny = 2*i
1480    GOSUB 50230
1490 :
1500    obenx  = 640-2*i
1510    untenx = 640-i
1520    GOSUB 50230
1530 :
1540    obeny  = 400-i
1550    unteny = 400-2*i
1560    GOSUB 50230
1570 :
1580    obenx  = i
1590    untenx = 2*i
1600    GOSUB 50230
1610 :
1620 NEXT
1630 :
1640 GOSUB 40000
1650 :
```

Programm 2.5: Das Programm, das die in Abb. 2.4 gezeigte Bildschirmausgabe bewirkt

```
1660 REM ---     Zweites Beispiel     ---
1670 :
1680 FOR i=0 TO PI STEP PI/8
1690 :
1700    obenx  = 320 + 150*SIN(i)
1710    obeny  = 200 + 100*COS(i)
1720    untenx = 320 + 300*SIN(i)
1730    unteny = 200 + 150*COS(i)
1740 :
1750    GOSUB 50230
1760 :
1770    obenx  = 320 - 300*SIN(i)
1780 REM  obeny  = 200 - 100*COS(i)
1790    untenx = 320 - 150*SIN(i)
1800 REM   unteny = 200 - 150*COS(i)
1810 :
1820    GOSUB 50230
1830 :
1840 NEXT
1850 :
1860 GOSUB 40000
1870 :
```

Programm 2.6: Ein weiteres Beispielprogramm für die Verwendung des Block-Unterprogramms

gleichen Vorgaben wie für die Rechtecke bei den Testdaten ergibt sich das Muster aus Abb. 2.4.

Das Listing zu Abb. 2.4 ist Programm 2.5.

Aus Dokumentationsgründen für dieses Buch wurde auf programmiertechnische Tricks verzichtet, und die Zeilen werden nochmals aufgeführt.

Ein zweites Beispiel für die Darstellung von Blöcken zeigt Programm 2.6.

Hier wollen wir im Kreis vorgehen und die Blöcke entsprechend den Koordinaten im Kreis ausdehnen bzw. zusammenstauchen. Die Werte 0 und 2*PI sind die Anfangs- und Endwerte für die trigonometrischen Funktionen SIN() und COS(). Wir rechnen hierbei im Bogenmaß; auf die Rechnung im Gradmaß werden wir später noch eingehen. Die Schrittweite im gesamten Kreis beträgt PI/8, womit 16 Blöcke dargestellt werden. Dem findigen Leser wird sicherlich schon aufgefallen sein, daß man – anders als beim Unterprogramm zur Darstellung eines Rechtecks – die Koordinaten beim Block-Unterprogramm nicht vertauschen darf (speziell die Koordinaten obenx und untenx), da sonst die FOR...NEXT-Schleife nicht korrekt durchlaufen wird. Aus diesem Grund muß noch für die eine Hälfte des Kreises der durch Multiplikation mit SIN() und COS() erhaltene Wert vom Kreismittelpunkt abgezogen werden.

Die Zeilen 1770 und 1790 sind also zur Darstellung der Blöcke im Ganzkreis unbedingt erforderlich. Da zunächst am Bildschirm immer der rechts liegende Block und dann der linke Block dargestellt wird, reicht es auch aus, die Pro-

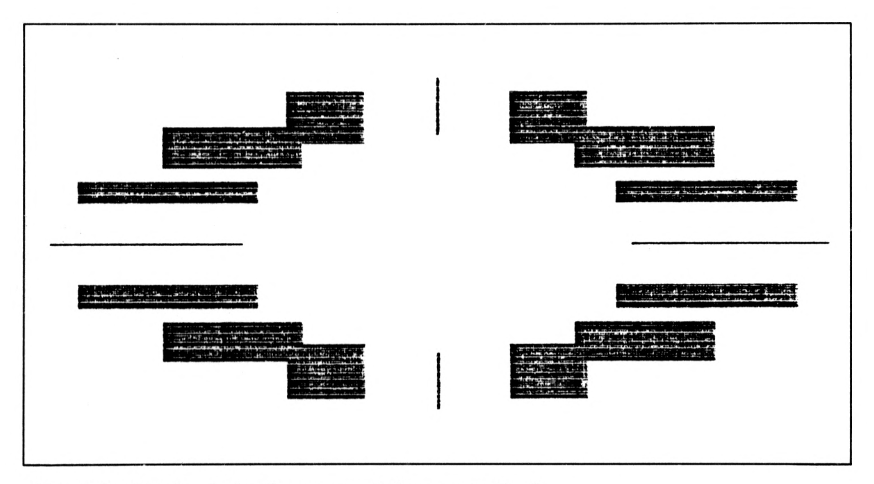


Abb. 2.5: Die durch das Programm 2.6 erzeugte Grafik

grammschleife von 0 bis PI laufen zu lassen. Auch Spezialfälle von Blöcken (Linien) sind sowohl senkrecht als auch waagerecht vorhanden.

Die in REM-Zeilen angegebenen Befehle in den Zeilen 1780 und 1800 sind für das Aussehen der Blöcke unerheblich, wie Sie leicht nachvollziehen können.

Die Ausgabe hat dann das in Abb. 2.5 gezeigte Aussehen.

Quader (leer)

Bleiben wir noch etwas bei den geradlinigen geometrischen Figuren. Eine Figur, deren grafische Darstellung auch in anderen Programmiersprachen nicht anzutreffen ist, ist der Quader, den wir zunächst als „Gerüst" darstellen wol-

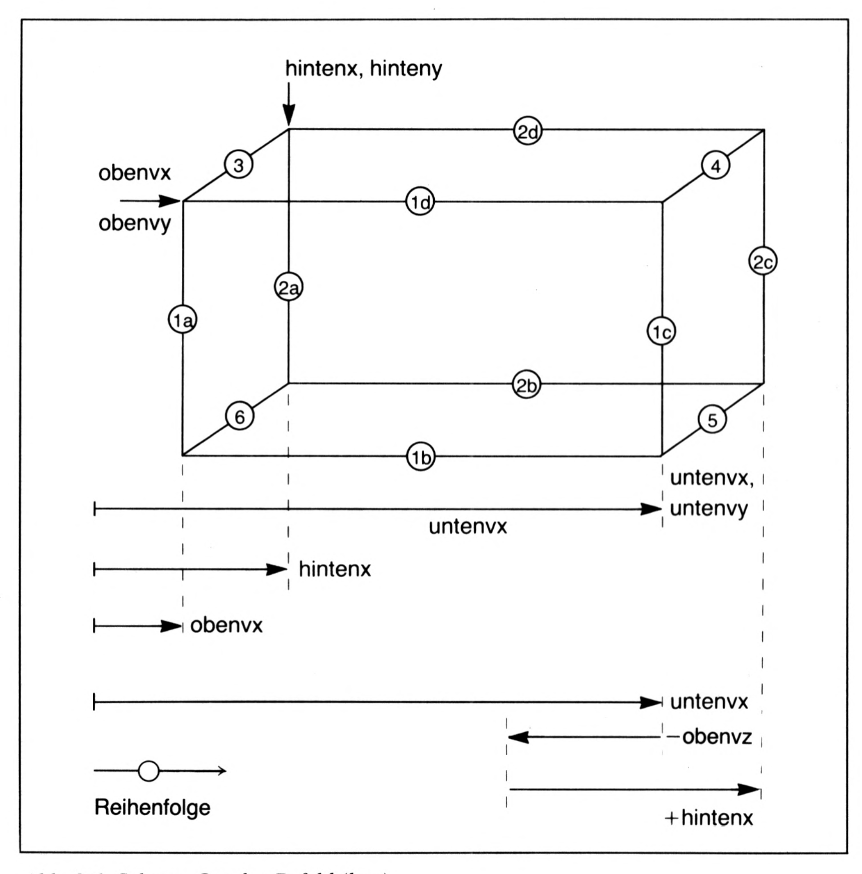


Abb. 2.6: Schema Quader-Befehl (leer)

len. Wenn man sich die zeichnerische Darstellung eines Quaders genau anschaut, stellt man fest, daß er aus einem vorderen und einem hinteren Rechteck besteht, wobei die Ecken der beiden Rechtecke jeweils paarweise miteinander verbunden werden.

Diesen Umstand wollen wir uns auch im folgenden zunutze machen und zunächst das vordere und hintere Rechteck mit dem bereits bekannten Unterprogramm ausgeben. Da wir für das hintere Rechteck spezielle Variablen brauchen, können wir die Variablen obenx, untenx, obeny und unteny nicht mehr übernehmen. Daher werden die Variablen obenvx, obenvy, untenvx, untenvy, hintenx und hinteny entsprechend Abb. 2.6 festgelegt, wobei die Variablen obenvx, obenvy, untenvx und untenvy sinngemäß zu den beiden schon beschriebenen Unterprogrammen verwendet werden und in ihrem Na-

```
52000 REM -----------------------------
52010 REM ---      Quader leer      ---
52020 REM -----------------------------
52030 :
52040 REM ---   vorderes Rechteck   ---
52050 :
52060    obenx  = obenvx
52070    obeny  = obenvy
52080    untenx = untenvx
52090    unteny = untenvy
52100    GOSUB 50120
52110 :
52120 REM ---   hinteres Rechteck   ---
52130 :
52140    obenx  = hintenx
52150    obeny  = hinteny
52160    untenx = untenvx+hintenx-obenvx
52170    unteny = untenvy+hinteny-obenvy
52180    GOSUB 50120
52190 :
52200 REM ---   restliche Linien    ---
52210 :
52220    MOVE obenvx,obenvy
52230    DRAW hintenx,hinteny,farbe
52240 :
52250    MOVE untenvx,obenvy
52260    DRAW untenvx+hintenx-obenvx,hinteny,farbe
52270 :
52280    MOVE untenvx,untenvy
52290    DRAW untenvx+hintenx-obenvx,untenvy+hinteny-ob
envy,farbe
52300 :
52310    MOVE obenvx,untenvy
52320    DRAW hintenx,untenvy+hinteny-obenvy,farbe
52330 :
52340 RETURN
52350 :
```

Programm 2.7: Das Quader-Unterprogramm

men lediglich ein v mehr ausweisen, wodurch dokumentiert ist, daß es sich um die vorderen Koordinaten handelt.

Für die räumliche Darstellung ist noch ein drittes Koordinatenpaar (hintenx, hinteny) erforderlich, womit aber die Größe und Position des Quaders ausreichend beschrieben ist. Alle anderen Ecken des Quaders ergeben sich aus den vorgenannten, wie auch aus Abb. 2.6 ersichtlich ist.

Um den für Nichtmathematiker doch recht komplizierten Sachverhalt zu veranschaulichen, haben wir den x-Teil der Koordinate „unten/rechts/hinten" durch Pfeile entsprechend der Länge (Vektoren) dargestellt. Die drei daran beteiligten Pfeile obenvx, untenvx und hintenx wurden gesondert mit ihrer Länge dargestellt, wobei ein Pfeil natürlich am Bildschirmrand beginnt. In diesem Fall gibt die Länge des Pfeils Auskunft über die Größe der Variablen. Wenn man nun die Pfeile geschickt aneinanderreiht, erhält man alle Ecken des Quaders. Für unsere Beispielecke müssen die Werte für untenvx und hintenx aufsummiert werden, wovon der Wert für obenvx abzuziehen ist.

Das Quader-Unterprogramm zeigt Programm 2.7.

Die einzelnen Teilbereiche zur Darstellung des Quaders sind deutlich zu erkennen. Ab Zeile 52060 werden zunächst die Variablen obenx, obeny, untenx und unteny mit den Werten des Quaders besetzt, und dann wird das Unterprogramm zum Zeichnen eines Rechtecks aufgerufen. Analog geschieht dies mit den Werten für das hintere Rechteck ab Zeile 52140. Dann werden ab Zeile 52220 die vier einzelnen Linien gezogen. Die Richtigkeit der Koordinaten können Sie an Abb. 2.6 nachvollziehen.

Ein Beispiel für die Anwendung des Quader-Unterprogramms sehen Sie in Programm 2.8.

```
1880 REM ------------------------------
1890 REM ---   Testdaten fuer Quader ---
1900 REM ---        (leer)           ---
1910 REM ------------------------------
1920 :
1930 obenvx  = 50
1940 obenvy  = 100
1950 untenvx = 150
1960 untenvy = 50
1970 hintenx = 60
1980 hinteny = 115
1990 :
2000 GOSUB 52000
2010 :
2020 obenvx  = 250
```

Programm 2.8: Anwendung des Quader-Unterprogramms

```
2030 obenvy  = 200
2040 untenvx = 200
2050 untenvy = 175
2060 hintenx = 280
2070 hinteny = 200
2080 :
2090 GOSUB 52000
2100 :
2110 obenvx  = 10
2120 obenvy  = 390
2130 untenvx = 300
2140 untenvy = 300
2150 hintenx = 40
2160 hinteny = 360
2170 :
2180 GOSUB 52000
2190 :
2200 GOSUB 40000
2210 :
2220 FOR i=10 TO 350 STEP 40
2230    obenvx  = i
2240    obenvy  = 400-i
2250    untenvx = 2*i
2260    untenvy = 400-1.1*i
2270    hintenx = i+30
2280    hinteny = 400-i+30
2290    GOSUB 52000
2300 NEXT
2310 :
2320 GOSUB 40000
2330 :
```

Programm 2.8: Anwendung des Quader-Unterprogramms (Forts.)

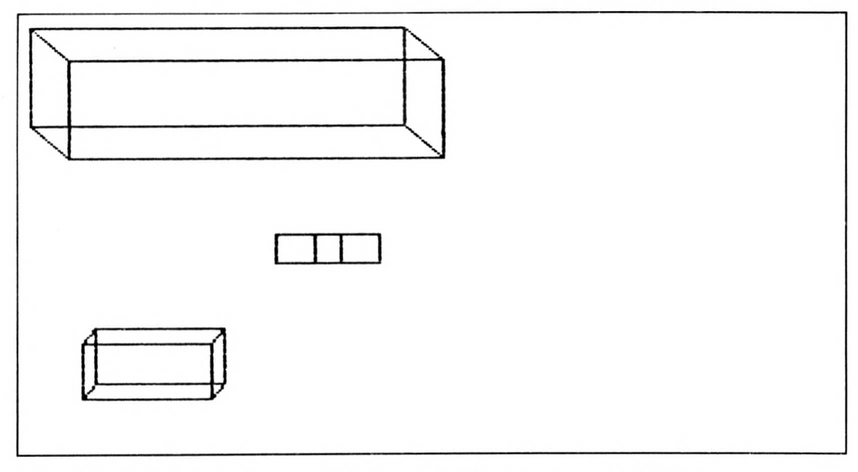


Abb. 2.7: Durch das Programm 2.8 erzeugte Bildschirmausgabe (Teil 1)

Das Ergebnis des ersten Beispielausdrucks ist aus Abb. 2.7 zu ersehen.

Zunächst wird der Quader unten links ausgegeben, was den „Normalfall“ für diese geometrische Figur darstellt. Dann wird der darüberliegende Quader ausgegeben, wodurch Sie sehen können, daß auch Spezialfälle möglich sind, die aus optischer Sicht keinen Quader ergeben. Der vorliegende „Quader“ kommt zustande, weil die Koordinaten obenvy und hinteny identisch sind. Das letzte Beispiel – oben links in der Abbildung – macht deutlich, daß die nach „hinten“ führenden Kanten nicht unbedingt von links nach rechts gezogen werden müssen. Da nur das Unterprogramm zum Zeichnen von Rechtekken und zusätzlichen Linien verwendet wird, ist es beim – leeren – Quader auch unerheblich, wie die Koordinaten auf die einzelnen Punkte verteilt sind.

Nachdem durch das Unterprogramm ab Zeile 40000 (Aufruf in Zeile 2200) der Bildschirm zum Schluß wieder gelöscht wurde, können wir einen weiteren Beispielausdruck starten, wie er in den Zeilen ab 2220 vorgegeben ist. Ähnlich den Beispielen bei Rechtecken und Blöcken werden auch hier sich jeweils vergrößernde Quader vorgestellt, die sich teilweise überlappen. Auch hier ist es zu empfehlen, mit den Variablen etwas herumzuprobieren. Durch einfache Änderung können sich wirkungsvolle Effekte ergeben. Abb. 2.8 zeigt das Ergebnis der Daten aus den Zeilen 2220 bis 2300.

Quader (ausgefüllt)

Ähnlich dem Übergang vom Rechteck zum Block wollen wir auch hier den Übergang vom Quadergerüst zum ausgefüllten Quader machen. Die Vorge-

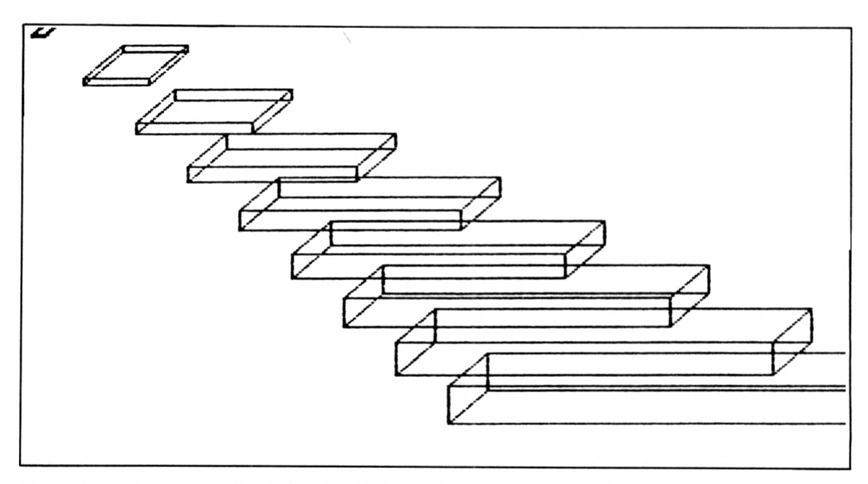


Abb. 2.8: Durch das Programm 2.8 erzeugte Bildschirmausgabe (Teil 2)

hensweise ist etwa analog, da wir auch hier vom einfachen Ziehen von Linien zum Zeichnen mittels FOR...NEXT-Schleifen übergehen müssen. Um Ihnen weitere Möglichkeiten zu bieten, wollen wir für die drei sichtbaren Seiten des ausgefüllten Quaders auch drei verschiedene Farben zulassen, die in den Variablen farbe1, farbe2 und farbe3 an das Unterprogramm übergeben werden müssen.

Wie aus dem ersten Teil des Listings ersichtlich ist, läßt sich die vordere Fläche des Quaders sehr einfach mit dem Block-Unterprogramm einfärben. Die zweite Fläche – normalerweise als „oben" bezeichnet – wird ebenso von links nach rechts eingefärbt und die dritte Fläche (rechte Seite) von unten nach oben. Die Laufvariable quader sollte im Hauptprogramm nicht verwendet werden.

Würde an dieser Stelle das Unterprogramm abschließen, so ergäbe sich ein Quader wie in Abb. 2.9 dargestellt.

Da das Ergebnis mehr einem entarteten zweiseitigen Pfeil ähnelt als einem Quader (jedenfalls immer bei einer Hardcopy, aber auch wenn farbe1 = far-

```
53000 REM -----------------------------
53010 REM ---      Quader voll      ---
53020 REM -----------------------------
53030 :
53040    obenx  = obenvx
53050    obeny  = obenvy
53060    untenx = untenvx
53070    unteny = untenvy
53080    farbe=farbe1
53090    GOSUB 51000
53100 :
53110    FOR quader=obenvx TO untenvx
53120       MOVE quader,obenvy
53130       DRAW quader+hintenx-obenx,hinteny,farbe2
53140    NEXT
53150 :
53160    FOR quader=untenvy TO obenvy
53170       MOVE untenvx,quader
53180       DRAW untenvx+hintenx-obenvx,quader+hinteny-
obenvy,farbe3
53190    NEXT
53200 :
53210    MOVE obenvx,obenvy
53220    DRAW untenvx,obenvy,0
53230    DRAW untenvx,untenvy,0
53240    MOVE untenvx,obenvy
53250    DRAW untenvx+hintenx-obenvx,hinteny,0
53260 :
53270 RETURN
53280 :
```

Programm 2.9: Das Vollquader-Unterprogramm

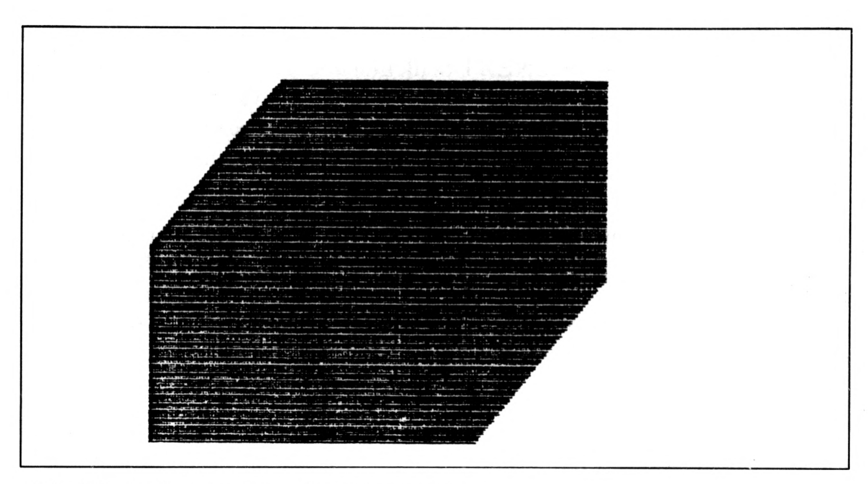


Abb. 2.9: Vollquader ohne Flächenbegrenzung

be2 = farbe3), wollen wir die Kanten noch nachziehen, wie es ab Zeile 53210 geschieht. Dies ist die Kante vorne oben, die Kante vorne rechts und die Kante oben rechts. Das Ergebnis läßt sich aus Abb. 2.10 ersehen.

Zum Schluß zeigen wir wieder die schematische Vorgehensweise beim ausgefüllten Quader (Abb. 2.11).

Das Listing für die Testdaten finden Sie in Programm 2.10.

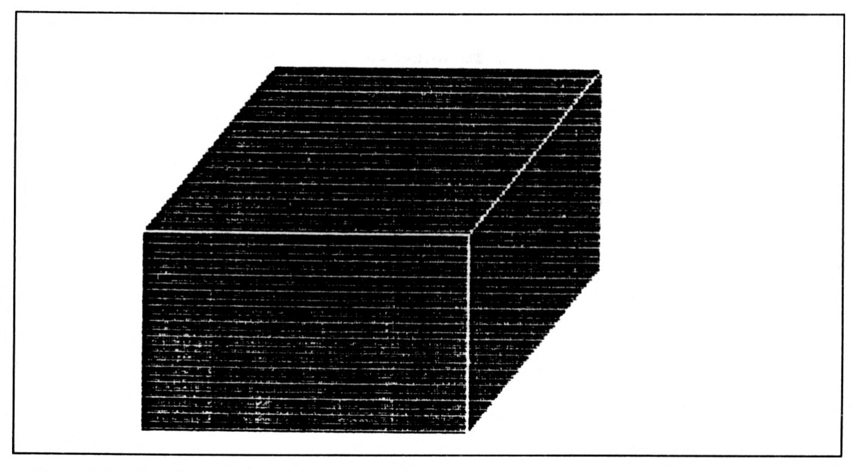


Abb. 2.10: Quader mit nachgezogenen Kanten

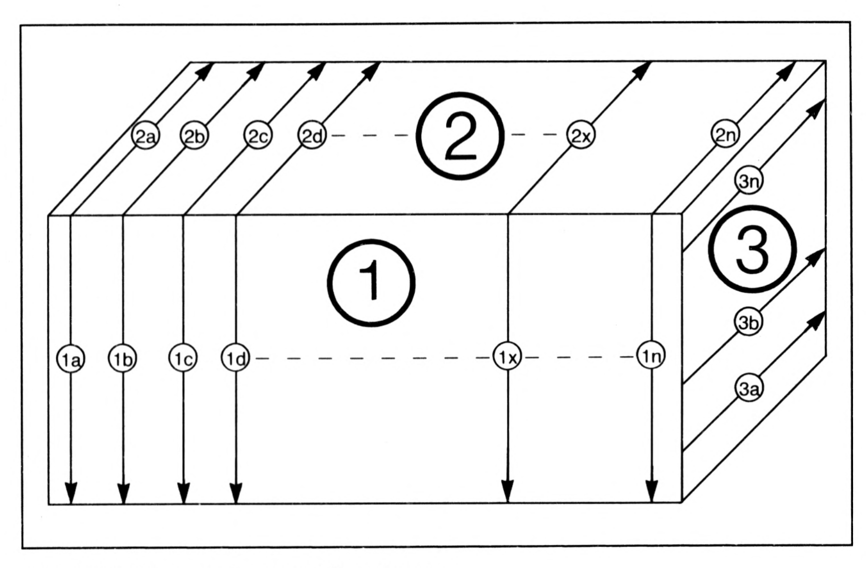


Abb. 2.11: Schema beim ausgefüllten Quader

```
2340 REM ------------------------------
2350 REM ---  Testdaten fuer Quader ---
2360 REM ---        (voll)          ---
2370 REM ------------------------------
2380 :
2390 obenvx  = 100
2400 obenvy  = 200
2410 untenvx = 350
2420 untenvy = 20
2430 hintenx = 200
2440 hinteny = 350
2450 :
2460 INK 3,4
2470 farbel = 1
2480 farbe2 = 2
2490 farbe3 = 3
2500 :
2510 GOSUB 53000
2520 :
```

Programm 2.10: Anwendung des Vollquader-Unterprogramms

Kreis, Ellipse und n-Eck

Verlassen wir nun die geraden geometrischen Figuren, und gehen wir zu den Kurven über. Als erstes steht da natürlich der Kreis zur Debatte, von dem sich

einige andere Fälle ableiten lassen. Die Schneider-Rechner stellen uns mit den Befehlen DEG und RAD die freie Auswahl zwischen Bogenmaß und Gradmaß zur Verfügung. Wer lieber mit Teilen oder Vielfachen der Zahl PI arbeitet, kann das Bogenmaß wählen; wer sich bei den Gradzahlen (0°–360°) besser auskennt, sollte den Befehl DEG voranstellen. Von Hause aus – nach dem Einschalten – ist der RAD-Modus eingestellt.

Da die Punkte eines Kreises einzeln gesetzt werden müssen, ist eine FOR...NEXT-Schleife unausweichlich. Zunächst: Welche Parameter werden gebraucht?

Die Schleifenparameter:

klauf – Laufvariable für die Kreisausgabe
kanf – Beginn des Kreises (Startwert für FOR...NEXT-Schleife)
kend – Ende des Kreises (Endekriterium der FOR...NEXT-Schleife)
kstep – Schrittweite (der FOR...NEXT-Schleife; siehe auch unten)

Mittelpunkt des Kreises:
mittelx; mittely

Radien des Kreises:
radiusx, radiusy

Setzen der Punkte:
punktx, punkty

Farbe des Kreises:
kfarbe

Für kanf und kend sowie kstep müssen jeweils die gewünschten Werte in Grad- oder Bogenmaß für Beginn und Ende angegeben werden. Durch kstep wird die Feinheit der grafischen Darstellung des Kreises direkt beeinflußt, so daß auch n-Ecke durch große Schrittweiten gezeichnet werden können (siehe folgende Beispiele). Durch die Verwendung zweier verschiedener Radien sind auch Ellipsen möglich (ein Kreis ist der Spezialfall einer Ellipse, wo beide Radien identisch sind).

Das Listing für das Kreis-Unterprogramm zeigt Programm 2.11.

Wie man sieht, ist das Unterprogramm unabhängig von der Art der Berechnung (Bogenmaß oder Gradmaß). Seine Verwendung ist lediglich von der Art der Eingaben abhängig und natürlich vom eingeschalteten Modus (DEG/RAD). Da die Angaben im Bogenmaß nach dem Einschalten erwartet werden, wollen wir dies auch als Normalzustand betrachten. Für Nichtmathematiker: Im Bogenmaß wird der Winkel nicht zwischen 0 und 360 angegeben, sondern zwischen 0 und 2*PI.

```
54000 REM ----------------------------
54010 REM ---   Kreis im Bogenmass   ---
54020 REM ----------------------------
54030 :
54040    FOR klauf=kanf TO kend STEP kstep
54050       punktx=mittelx+radiusx*SIN(klauf)
54060       punkty=mittely+radiusy*COS(klauf)
54070       IF klauf=kanf                          T
HEN MOVE punktx,punkty :                     GOTO 54090
54080       DRAW punktx,punkty,kfarbe
54085       winkel=klauf : GOSUB 56090
54090    NEXT
54100 :
54110 RETURN
54120 :
```

Programm 2.11: Das Kreis-Unterprogramm

Um auch im Gradmaß rechnen zu können, verwenden wir das kleine Unterprogramm 2.12.

Hierdurch wird zunächst im Rechner intern auf Rechnung im Gradmaß umgeschaltet, dann das zuletzt genannte Unterprogramm aufgerufen und anschließend wieder auf den definierten Normalzustand (Bogenmaß) zurückgeschaltet.

Besonders mit dem Unterprogramm zum Zeichnen von Kreisen (und natürlich auch Ellipsen und n-Ecken) lassen sich sehr schöne Grafiken erzeugen. Ein weiterer Vorteil ist die Möglichkeit, Beginn und Ende eines Kreises als Parameter mit anzugeben, wodurch auch Teilbereiche von Kreisen ausgegeben werden können. Schauen wir uns nun ein paar Beispiele an.

Zunächst wollen wir einen „normalen" Kreis ausgeben. Die Ausgabe erfolgt beim ersten Beispiel im Bogenmaß. Zunächst werden die Schleifenparameter

```
55000 REM ----------------------------
55010 REM ---   Kreis im Gradmass    ---
55020 REM ----------------------------
55030 :
55040    DEG
55050 :
55060    GOSUB 54000
55070 :
55080    RAD
55090 :
55100 RETURN
55110 :
```

Programm 2.12: Unterprogramm für die Rechnung im Gradmaß

```
2550 REM ------------------------------
2560 REM ---  Testdaten fuer Kreis  ---
2570 REM ---      im Bogenmass      ---
2580 REM ------------------------------
2590 :
2600 kanf    = 0
2610 kend    = 2*PI+2*PI/75
2620 kstep   = 2*PI/75
2630 mittelx = 320
2640 mittely = 200
2650 radiusx = 100
2660 radiusy = 100
2670 kfarbe  = 1
2680 :
2690 GOSUB 54000
2700 :
2710 GOSUB 40000
2720 :
```

Programm 2.13: Kreis im Bogenmaß

gesetzt, wobei der Kreis in 75 Teilschritten ausgegeben wird. Den Mittelpunkt des Kreises legen wir auf den Bildschirmmittelpunkt, und der Radius soll in beide Richtungen 100 Bildschirmpunkte betragen. Dann wird das Unterprogramm ab Zeile 40000 aufgerufen und das Unterprogramm zur Ausgabe einer Hardcopy (Programm 2.13). Das Ergebnis sehen Sie in Abb. 2.12.

Besonders im Falle des Kreises sei Ihnen nahegelegt, mit den Parametern zu experimentieren. Ellipsen ergeben sich z. B. durch unterschiedliche Werte in radiusx und radiusy.

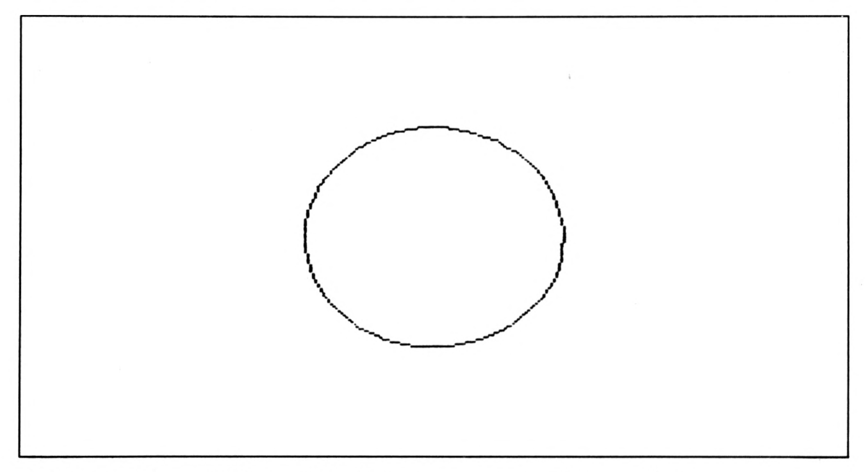


Abb. 2.12: Ein mit Hilfe des Kreis-Unterprogramms gezeichneter Kreis

```
2730 REM ------------------------------
2740 REM ---  Testdaten fuer Kreis   ---
2750 REM ---        im Gradmass      ---
2760 REM ------------------------------
2770 :
2780 kend    = 360
2790 kstep   = 10
2800 :
2810 GOSUB 55000
2820 :
2830 GOSUB 40000
2840 :
```

Programm 2.14: Kreis im Gradmaß

Das gleiche Erscheinungsbild erhalten wir durch die in dem Programm 2.14 benutzten Daten.

Da wir Mittelpunkt und Radius beibehalten wollen und der Anfang des Kreises bereits auf 0 Grad festgelegt ist (bei der Null ist es egal, ob sie im Bogenmaß oder im Gradmaß angesehen wird), brauchen wir nur noch die Schrittweite und das Ende des Kreises im Gradmaß festzulegen. Natürlich muß auch das Unterprogramm zum Umschalten auf DEG aufgerufen werden.

Mathematiker mögen mir den folgenden Programmtitel verzeihen (es ist sogar bewiesen, daß die Quadratur des Kreises mit Zirkel und Lineal unmöglich ist).

```
2850 REM ------------------------------
2860 REM -'Die Quadratur des Kreises'--
2870 REM ------------------------------
2880 :
2890 kanf=PI/4
2900 kend=2*PI+PI
2910 :
2920 FOR j=15 TO 2 STEP -1
2930    kstep = PI/j
2940    GOSUB 54000
2950    FOR i=1 TO 1000
2960    NEXT
2970    INK 1,0
2980    kfarbe= 0
2990 REM GOSUB 50000
3000    INK 1,26
3010    kfarbe= 1
3020 NEXT
3030 :
3040 GOSUB 40000
3050 :
```

Programm 2.15: „Quadratur" des Kreises

Außerdem ist ein Kriterium der bekannten mathematischen Frage nicht eingehalten: Kreis und Quadrat sind nicht flächengleich. Der Kreis ist vielmehr der Umkreis zum letztendlich erscheinenden Quadrat.

Der Programmschleife im Unterprogramm zur Darstellung eines Kreises wird in den Testdaten ab Zeile 2920 eine weitere Programmschleife vorgelagert, die die Schrittweite bei der Darstellung des Kreises jeweils vermindert. Das 30-Eck wird nach und nach zu einem Viereck (Quadrat) umgeformt. Das 30-Eck ist dabei kaum von einem Kreis zu unterscheiden.

An dieser Stelle wird auch deutlich, daß die grafische Darstellung eines Kreises mittels eines Computers am Bildschirm durch die Bildschirmpunkte nur die Annäherung an einen Kreis sein kann.

Dadurch, daß nicht einzelne Punkte eines Kreises in dem Unterprogramm ab Zeile 54000 ausgegeben werden, sondern Linien vom letzten errechneten Punkt zum aktuellen Punkt (Zeile 54080) ist es möglich, auf diese einfache Weise n-Ecke auszugeben. Das überlagerte Ergebnis aller Ausgaben aus der Programmschleife zwischen den Zeilen 2920 und 3020 zeigt Abb. 2.13.

Sofern Sie sich die einzelnen Schritte getrennt ausgeben lassen, werden Sie feststellen, daß schon ein 15-Eck eine gute Annäherung an einen Kreis darstellt. Dies ist besonders dann wichtig, wenn das Zeichnen eines Kreises möglichst wenig Rechenzeit kosten soll.

Wie bereits erwähnt, ist es möglich, durch die Werte kanf und kend Ausschnit-

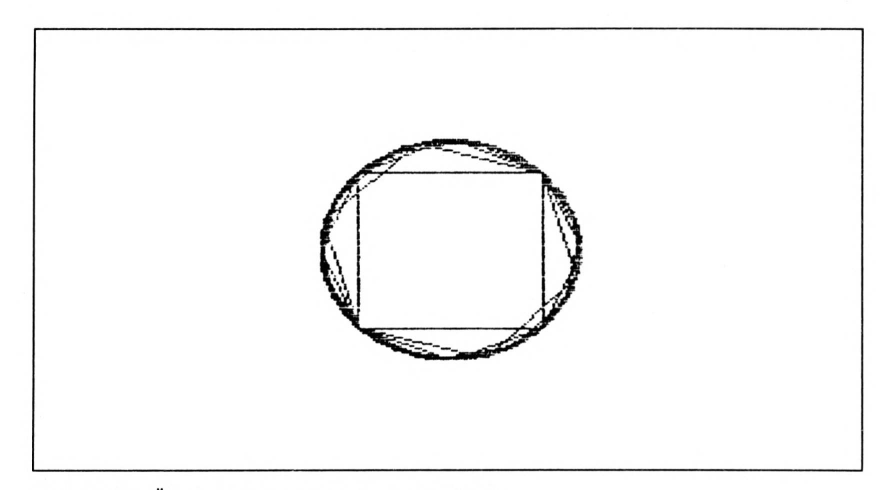


Abb. 2.13: Übergang vom „Kreis" zum Rechteck

```
3060 REM ------------------------------
3070 REM -- Beispiele mit Teilkreisen -
3080 REM ------------------------------
3090 :
3100 mittelx = 50
3110 mittely = 50
3120 kanf    = 90
3130 kend    = 360
3140 kstep   = 5
3150 :
3160 FOR i=5 TO 50 STEP 5
3170    radiusx = i+20
3180    radiusy = i
3190    GOSUB 55000
3200 NEXT
3210 :
3220 mittelx = 145
3230 kanf    = 270
3240 kend    = 360
3250 :
3260 FOR i=5 TO 50 STEP 5
3270    radiusx = i+20
3280    radiusy = i*5
3290    GOSUB 55000
3300 NEXT
3310 :
3320 mittely = 400
3330 kanf    = 90
3340 kend    = 180
3350 :
3360 FOR i=325 TO 100 STEP -25
3370    radiusx = i*2
3380    radiusy = i
3390    GOSUB 55000
3400 NEXT
3410 :
3420 mittely = 0
3430 kanf    = 0
3440 kend    = 90
3450 :
3460 FOR i=75 TO 300 STEP 25
3470    radiusx = i*3
3480    radiusy = i
3490    GOSUB 55000
3500 NEXT
3510 :
3520 GOSUB 40000
3530 :
```

Programm 2.16: Beispiele für Teilkreise

te eines Kreises auszugeben, was wir im folgenden Beispiel verwenden wollen (Programm 2.16).

Mit dem bisher vorhandenen Wissen ist es eine gute Übung für Sie herauszufinden, welcher Teil der Abb. 2.14 an welcher Stelle im Programm initiiert wird.

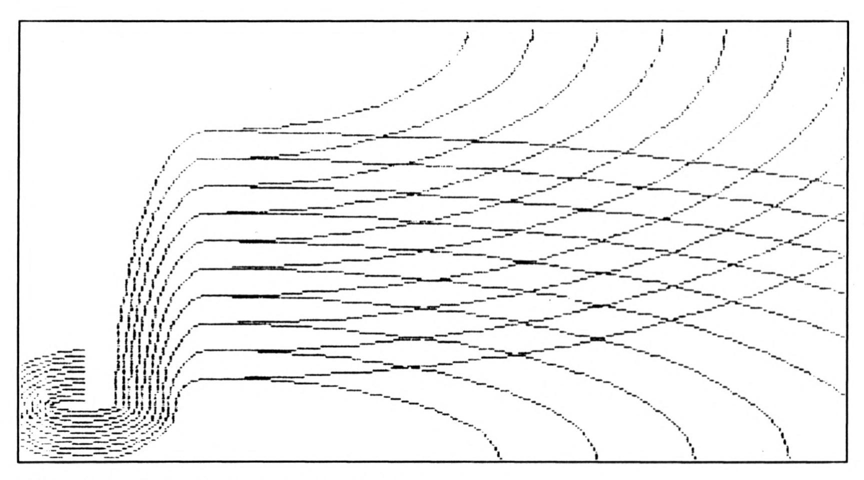


Abb. 2.14: Ergebnis des Programms 2.16

Spiralen

Mit dem Unterprogramm zur Darstellung eines Kreises können auch andere Formen dargestellt werden, wie z. B. Spiralen. Zunächst wollen wir das Un-

```
3540 REM ------------------------------
3550 REM ---     Spiralen           ---
3560 REM ------------------------------
3570 :
3580 MODE 2
3590 :
3600 mittelx = 320
3610 mittely = 200
3620 radiusx = 0
3630 radiusy = 0
3640 :
3650 FOR i=0 TO 14400 STEP 10
3660    winkel = i MOD 360
3670    radiusx = radiusx+0.2
3680    radiusy = radiusy+0.2
3690    kanf    = winkelalt
3700    IF kanf = 360 THEN kanf = 0
3710    kend    = winkel
3720    IF kend = 0 THEN kend = 360
3730    winkelalt = winkel
3740    kstep = 10
3750    GOSUB 55000
3760 NEXT
3770 :
3780 GOSUB 40000
3790 :
```

Programm 2.17: Zeichnen von Spiralen

terprogramm in seiner ursprünglichen Form verwenden und geben dazu die in Programm 2.17 gezeigten Testdaten ein.

Um die Genauigkeit der Darstellung zu erhöhen, schalten wir den Bildschirmmodus auf 2 um. Der Mittelpunkt unserer Spirale soll auch im Mittelpunkt des Bildschirms liegen, die beiden Radien sind zu Beginn 0. Die Berechnung der Spiralen erfolgt im Gradmaß, wobei wir die Modulo-Funktion verwenden, die im Handbuch nur unzureichend beschrieben ist. Mit dieser Modulo-Funktion (x MOD y) wird von dem Wert vor der Funktion (x) so oft der andere Wert (y) abgezogen, bis kein Subtrahieren mehr möglich ist. Das Ergebnis ist der verbleibende Rest.

In unserem Fall erhalten wir so in Zeile 3660 Werte zwischen 0 und 360 Grad, wie wir sie für die Beschreibung eines Kreises im Gradmaß benötigen. Innerhalb der Schleife von Zeile 3650 bis Zeile 3760 wird dann jeweils der Radius um 0,2 Bildpunkte erhöht. Da die Schrittweite der Programmschleife 10 beträgt, werden bis zum gesamten Umlauf 36 mal 0,2 addiert, was einen Abstand bei „benachbarten" Bildschirmpunkten von neun Einheiten ausmacht. Damit nur ein Teil eines Kreises im Unterprogramm ab Zeile 55000 ausgegeben wird, findet in den Zeilen 3690 und 3710 die Vorbesetzung der betreffenden Variablen statt, wobei jeweils vom alten Winkel ausgehend bis zum neuen Wert der Kreisbogen gezogen wird.

Da es durch Verwendung der Programmschleife im Unterprogramm zum Ausgeben eines Kreises zu Konflikten mit den Anfangs- und Endwerten kommen kann, wird in den Zeilen 3700 und 3720 jeweils noch der Wert für den

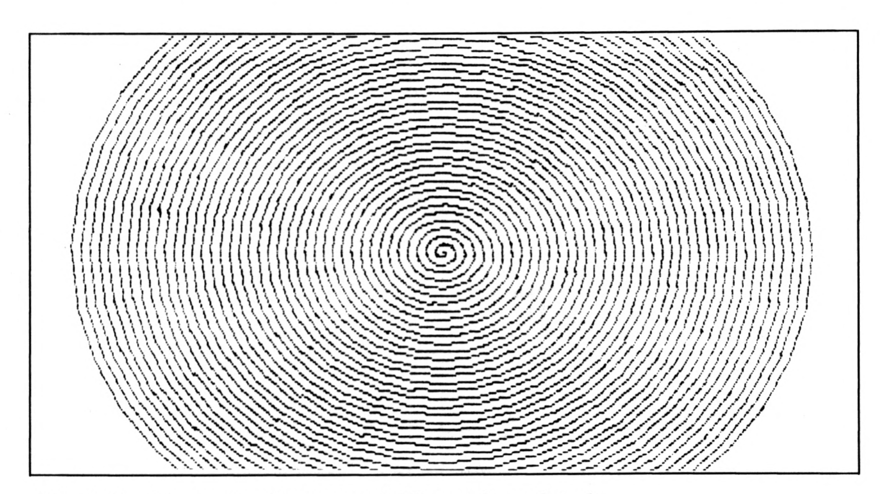


Abb. 2.15: Die von dem Programm 2.17 gezeichnete Spirale

Kreisanfang bzw. das Kreisende entsprechend festgelegt. Ohne diesen Trick müßte das Unterprogramm entsprechend geändert werden, oder aber Teilbereiche der Spirale würden nicht gezeichnet. Für den nächsten Durchgang wird dann noch die Variable winkelalt mit dem aktuellen Winkel besetzt und zum Schluß der Programmschleife das Unterprogramm zur Ausgabe des Kreisausschnittes aufgerufen.

Das Ergebnis sehen Sie in Abb. 2.15.

Da bei jedem neuen Ansatz zur Ausgabe eines Teilkreises der Radius sprunghaft nach außen verschoben wird, kann man objektiv gesehen die Spirale nicht als durchgehende Linie betrachten. Bei genügend großer Schrittweite wird dies auch am Bildschirm und in der Hardcopy offensichtlich.

Aus diesem Grunde wollen wir zum Thema Spirale noch eine bessere Möglichkeit vorstellen.

Das Ergebnis zu den vorgegebenen Daten ist in Abb. 2.16 festgehalten.

Auch hier beginnen wir zunächst bei beiden Radien mit Nullwerten. Da wir das Unterprogramm ab Zeile 55000 nicht aufrufen wollen, müssen wir zu-

```
3800 REM ------------------------------
3810 REM ---    Bessere Spiralen     ---
3820 REM ------------------------------
3830 :
3840 radiusx = 0
3850 radiusy = 0
3860 :
3870 DEG
3880 :
3890 MOVE mittelx,mittely
3900 :
3910 FOR i=0 TO 9000 STEP 28
3920    winkel = i MOD 360
3930    radiusx = radiusx+1
3940    radiusy = radiusy+0.6
3950    punktx  = mittelx+radiusx*SIN(winkel)
3960    punkty  = mittely+radiusy*COS(winkel)
3970    DRAW punktx,punkty,1
3980 GOTO 4010
3990    DRAW mittelx,mittely,1
4000    MOVE punktx,punkty
4010 NEXT
4020 :
4030 REM RAD
4040 :
4050 GOSUB 40000
4060 :
```

Programm 2.18: Verbessertes Spiral-Programm

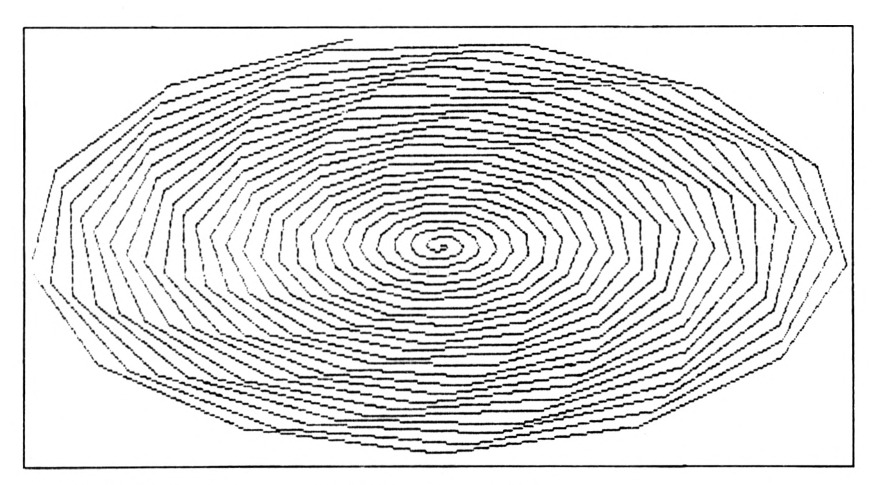


Abb. 2.16: Die von Programm 2.18 gezeichnete Spirale

nächst selbst den Rechenmodus auf DEG umschalten. Dann setzen wir den Grafik-Cursor auf den Mittelpunkt der Spirale zurück, da er vom letzten Beispiel eine unerwünschte Position hat.

Es wurde mit Absicht eine hohe Schrittweite gewählt (28°), um Ihnen auch einmal die künstlerischen Effekte vorzuführen. Versuchen Sie die gleiche Schrittweite bei unserem Beispiel ab Zeile 3540. Der Unterschied wird offensichtlich.

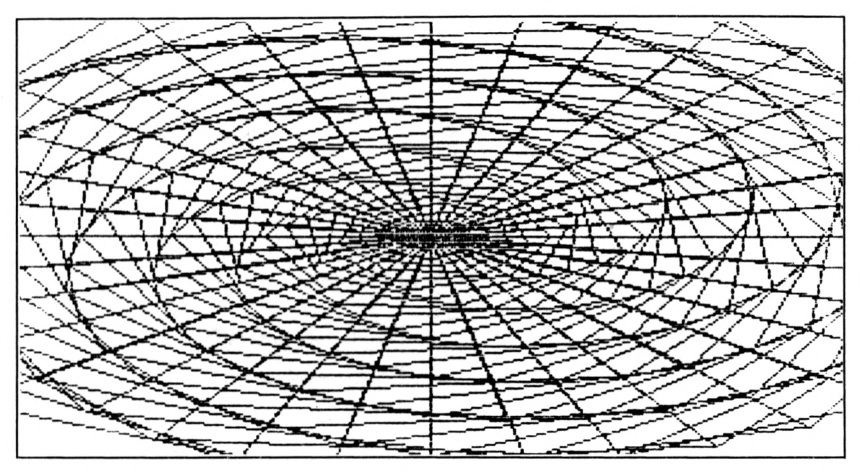


Abb. 2.17: Spirale mit Radien, Winkel 50°

Zunächst ziehen wir die Spirale in x-Richtung weiter auseinander als in y-Richtung (Zeilen 3930 und 3940). Dann wird der aktuelle Punkt bestimmt, zu dem eine Linie gezogen werden soll. Wenn Sie das GOTO 4010 in Zeile 3980 weglassen, wird außerdem noch vom aktuellen Punkt zum Mittelpunkt der Spirale eine Linie gezogen und der Grafik-Cursor wieder an die aktuelle Position gesetzt.

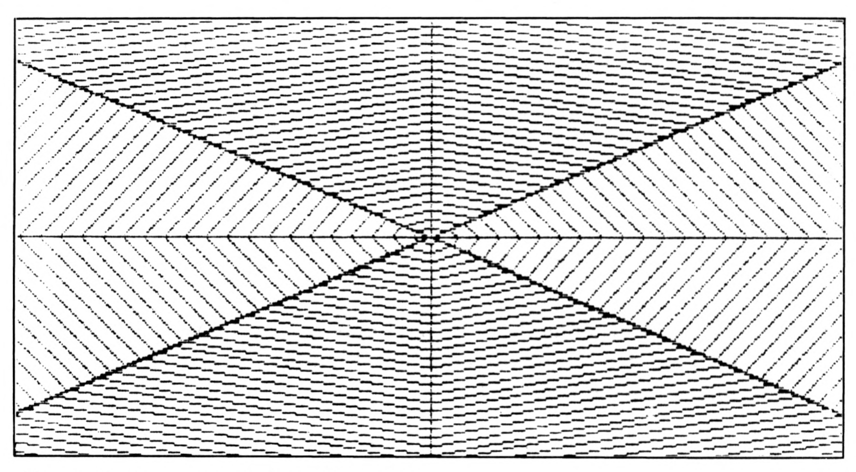


Abb. 2.18: Spirale mit Radien, Winkel 45°

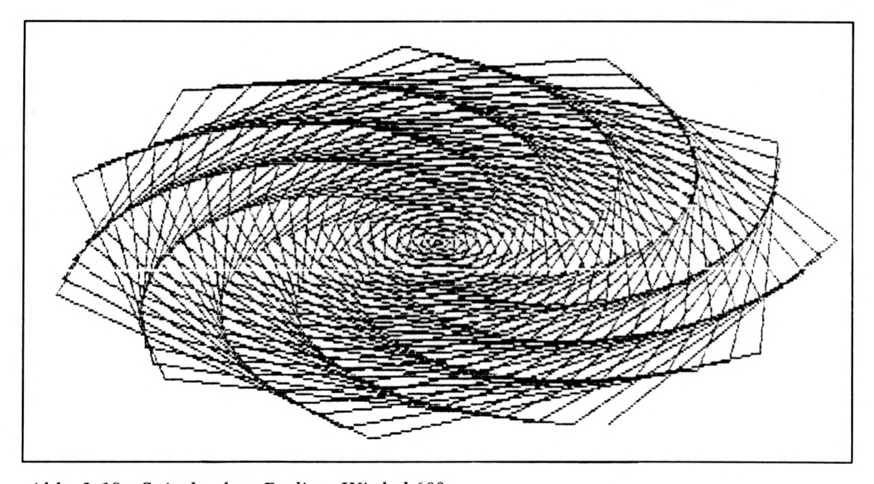


Abb. 2.19: Spirale ohne Radien, Winkel 60°

Abschließend wird wieder auf Bogenmaß umgeschaltet (Zeile 4030) und das Unterprogramm zur Ausgabe der Hardcopy aufgerufen.

Durch Ändern einiger weniger Parameter lassen sich leicht andere Effekte erreichen, wie Abb. 2.17 zeigt.

Behalten Sie die Ausgabe der Radien bei, und ändern Sie den Winkel in 45 Grad, so erhalten Sie die Ausgabe aus Abb. 2.18.

An dieser Stelle haben wir bewußt mehrere Ausdrucke abgebildet, um Ihnen die Vielfalt der Möglichkeiten trotz geringer Änderungen darzustellen. Abschließend noch ein weiteres Beispiel: Winkel 60 Grad; ohne Radien Abb. 2.19.

Radius

Natürlich braucht man nicht nur Kreise und Kreisbogen, sondern auch – wie wir im Kapitel über Tortendiagramme sehen werden – Verbindungslinien vom Kreismittelpunkt zum Kreisumfang. Dies wird normalerweise als Radius bezeichnet (Programm 2.19).

Hier wurden beide Möglichkeiten mittels Zahlangaben (Gradmaß/Bogenmaß) in ein einziges Unterprogramm gepackt, das nun mit verschiedenen Einsprungadressen aufgerufen werden kann. Da wir davon ausgehen können, daß im Normalfall das Bogenmaß eingeschaltet ist, braucht hierfür keine Umstellung zu erfolgen, und die Einsprungadresse ist die Zeile 56090. Sollen die Berechnungen im Gradmaß erfolgen, so ist als Einsprungadresse Zeile 56070 zu wählen, wo zunächst die Umschaltung erfolgt.

```
56000 REM ----------------------------
56010 REM ---         Radius       ---
56020 REM ---                      ---
56030 REM --- Einsprung Grad:  56070---
56040 REM --- Einsprung Bogen: 56090---
56050 REM ----------------------------
56060 :
56070    DEG
56080 :
56090    MOVE mittelx,mittely
56100 :
56110    DRAW mittelx+radiusx*SIN(winkel),mittely+radiu
sy*COS(winkel),farbe
56120 :
56125 GOTO 56150
56130    RAD
56140 :
56150 RETURN
56160 :
```

Programm 2.19: Zeichnen von Radien

Im ausführenden Teil selbst wird zunächst der Grafik-Cursor auf den Mittelpunkt des – in manchen Fällen imaginären – Kreisumfangs gesetzt. Dann wird mit den schon bekannten Winkelfunktionen Sinus und Cosinus der gewünschte Punkt auf dem Kreisumfang ermittelt, zu dem die Linie gezogen werden soll.

In Zeile 56125 ist noch ein Sprungbefehl eingefügt, der die Umschaltung auf das Bogenmaß überspringt. Dies ist wichtig, wenn z. B. im Hauptprogramm Berechnungen im Gradmaß durchgeführt werden sollen.

Selbstverständlich haben wir in diesem Fall auch wieder einige Testdaten vorbereitet (Programm 2.20).

```
4070 REM ------------------------------
4080 REM ---  Testdaten fuer Radius ---
4090 REM ------------------------------
4100 :
4110 FOR winkel=0 TO 720 STEP 3
4120    mittelx = winkel
4130    mittely = winkel/2
4140    radiusx = winkel/2
4150    radiusy = winkel/3
4160    GOSUB 56070
4170 NEXT
4180 :
4190 GOSUB 40000
4200 :
4210 FOR i=0 TO 20 STEP 5
4220    mittelx = 320+160*SIN(i)
4230    mittely = 200+100*COS(i)
4240    FOR winkel=0 TO 360 STEP 15
4250       radiusx = i+winkel/2
4260       radiusy = (i+winkel)/2
4270       GOSUB 56070
4280    NEXT
4290 NEXT
4300 :
4310 GOSUB 40000
4320 :
4330 mittelx = 320
4340 mittely = 200
4350 radiusx = 250
4360 radiusy = 150
4370 kanf    = 0
4380 kend    = 360
4390 kstep   = 20
4400 kfarbe  = 1
4410 :
4420 GOSUB 55000
4430 :
4440 GOSUB 40000
4450 :
```

Programm 2.20: Anwendung des Radius-Unterprogramms

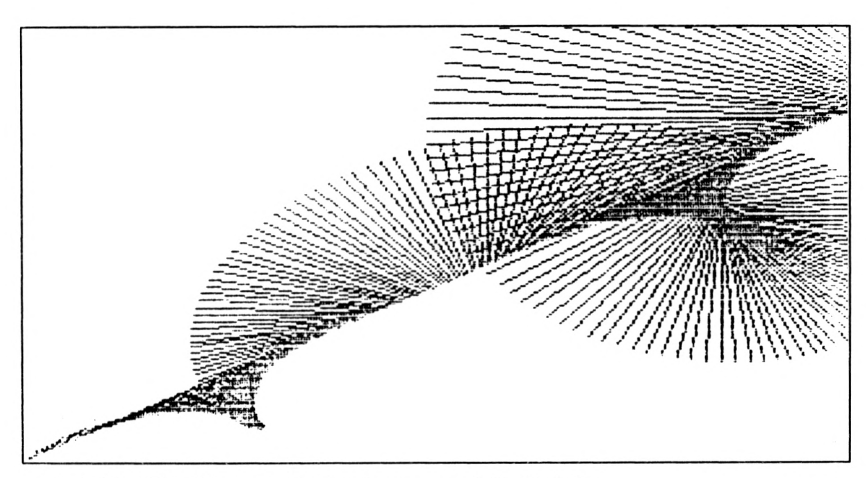


Abb. 2.20: Bildschirmausgabe des Programms 2.20 (Teil 1)

Im ersten Test (Zeilen 4110 bis 4170) werden zwei Vollkreise mit Schrittweite 3 durchlaufen. Da die Angaben im Gradmaß gegeben sind, ergibt sich als jeweiliger Mittelpunkt eine Linie, wie sie aus Abb. 2.20 ersichtlich ist.

Die Laufvariable winkel ist gleichzeitig Eingabeparameter für das Unterprogramm. Mit den Werten radiusx und radiusy wird jeweils die Länge der Radien beeinflußt, so daß sie mit größerer Anzahl von Durchläufen durch die Schleife jedesmal größer werden.

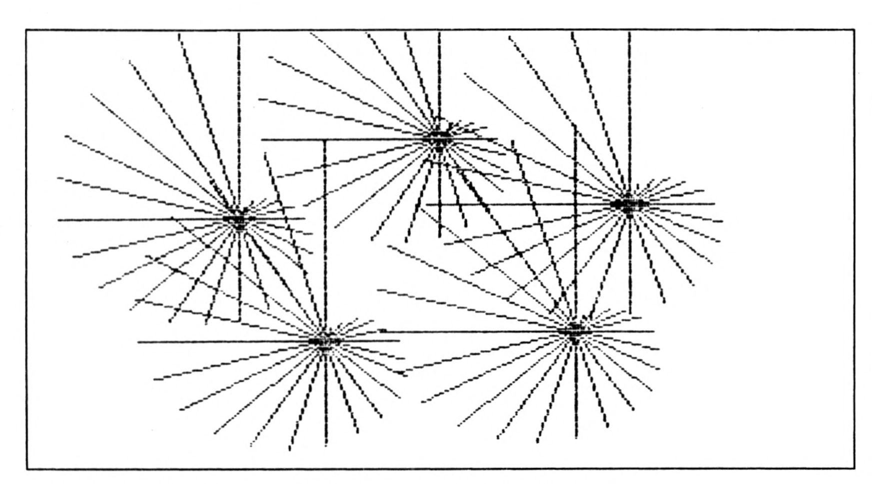


Abb. 2.21: Bildschirmausgabe des Programms 2.20 (Teil 2)

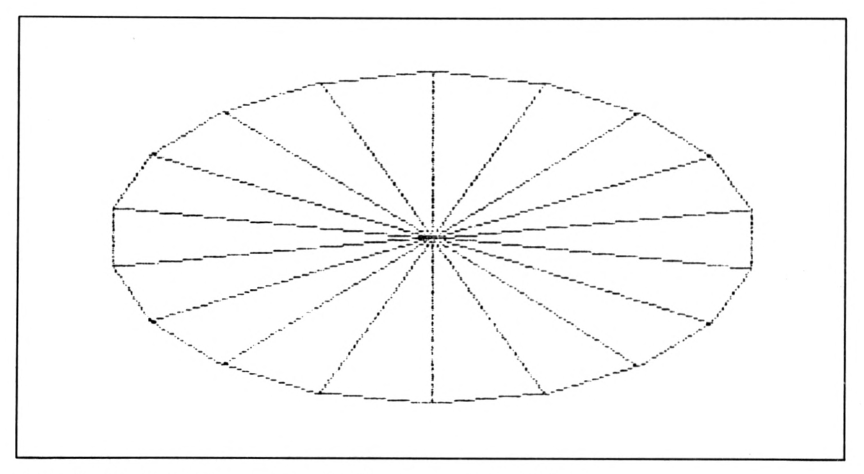


Abb. 2.22: Bildschirmausgabe des Programms 2.20 (Teil 3)

In Abb. 2.21 (Testdaten in den Zeilen 4210 bis 4290) wird in einer äußeren Schleife (Laufvariable i) jeweils der Mittelpunkt für die Radien festgelegt, und in einer weiteren Schleife (Laufvariable winkel) werden jeweils Radien mit 15 Grad Abstand ausgegeben, deren Länge sowohl in X- als auch Y-Richtung jeweils der Gradanzahl entspricht.

In Abb. 2.22 ist ein 18-Eck ausgegeben, in dem für jede Ecke die Radien eingezeichnet sind. Die Testdaten zu dieser geometrischen Figur befinden sich in den Zeilen 4330 bis 4400. Dafür wurde die Zeile 54085 in das Unterprogramm ab Zeile 54000 (Kreis im Bogenmaß) eingesetzt, wo der Eingabeparameter winkel für das Unterprogramm zum Zeichnen eines Radius mit dem aktuellen Wert der Laufvariablen klauf zum Zeichnen des Kreises besetzt wird und anschließend das Unterprogramm zum Ausgeben des Radius aufgerufen wird.

Besonders hier wird also das Zusammenspiel der einzelnen Unterprogramme direkt deutlich.

Nun wollen wir noch einen weiteren Spezialfall des Kreises – programmtechnisch gesehen – besprechen.

Stern

Grafisch gesehen ist ein Stern nichts anderes als ein n-Eck, dem ein Kreis mit zwei verschiedenen Radien zugrunde liegt. Beim Umlauf des Kreises werden abwechselnd die zu den jeweiligen Winkeln gehörigen Punkte der beiden verschiedenen Radien miteinander verbunden. Dies dürfte auch für Sie nicht wei-

ter schwierig sein, da wir ja das n-Eck schon als Sonderform – programmtechnisch gesehen – des Kreises kennengelernt haben.

Den breits bekannten Parametern für das Unterprogramm zum Zeichnen eines Kreises fügen wir also noch die Parameter radiusx1, radiusy1, radiusx2 und radiusy2 hinzu. Die ursprünglichen Radien werden – mit anderer Bedeutung – weiterverwendet. Damit ergibt sich das im Programm 2.21 gezeigte Unterprogramm.

Der MOVE-Befehl in Zeile 57040 kann gegebenenfalls auch weggelassen werden. In der schon bekannten Programmschleife mit den Parametern klauf, kanf und kend sowie kstep werden wieder die einzelnen Winkel berechnet. In den Zeilen 57070 und 57080 wird festgelegt, welche Radien für die nächste Ausgabe Gültigkeit haben. Die Hilfsvariable hilf kann von Ihnen vor dem Aufrufen auch gesetzt werden, um entweder mit dem engeren oder dem weiteren Radius zu beginnen.

Es ist unbedingt darauf zu achten, daß sich eine gerade Anzahl von Schritten ergibt, da sich sonst vielleicht unerwünschte Effekte einstellen – die allerdings auch ganz reizvoll sein können.

In den Zeilen 57090 und 57100 wird der entsprechende Punkt auf dem Kreisumfang ermittelt. Die Variablen radiusx und radiusy sind also in diesem Falle Hilfsvariablen des Unterprogramms und keine Eingabeparameter.

```
57000 REM ------------------------------
57010 REM ---           Stern         ---
57020 REM ------------------------------
57030 :
57040    MOVE mittelx,mittely
57050 :
57060    FOR klauf=kanf TO kend STEP kstep
57070       IF hilf=0 THEN radiusx=radiusx1 : radiusy=r
adiusy1 : hilf=1 : GOTO 57090
57080       IF hilf=1 THEN radiusx=radiusx2 : radiusy=r
adiusy2 : hilf=0
57090       punktx = mittelx+radiusx*SIN(klauf)
57100       punkty = mittely+radiusy*COS(klauf)
57110       IF erstlauf=0 THEN MOVE punktx,punkty : ers
tlauf=1 : GOTO 57130
57120       DRAW punktx,punkty,kfarbe
57130    NEXT
57140 :
57150    erstlauf = 0
57160    hilf     = 0
57170 :
57180 RETURN
57190 :
```

Programm 2.21: Unterprogramm zum Zeichnen eines Sterns

In der Zeile 57110 wird noch die Variable erstlauf geprüft, da das Zeichnen des Sterns natürlich nicht im Mittelpunkt beginnen soll. Nach dem Zeichnen des gesamten Sterns werden die Hilfsvariablen erstlauf und hilf wieder auf 0 zurückgesetzt, um bei einem weiteren Aufruf – z. B. einer Programmschleife im Hauptprogramm – von definierten Parametern ausgehen zu können. Es ist unbedingt darauf zu achten, daß die Hilfsvariablen radiusx, radiusy, erstlauf und hilf an anderer Stelle nicht im Hauptprogramm verwendet werden.

Nun kommen wir zur Anwendung des Stern-Unterprogramms (Programm 2.22).

Zunächst geben wir einen einfachen 15zackigen Stern aus (Abb. 2.23), was natürlich 30 Schritte beim Umlauf des Kreises zur Voraussetzung hat, daher die Schrittweite 12 (360/12 = 30). Die Testdaten dazu finden Sie in den Zeilen 4500 bis 4580.

Beim nächsten Testausdruck wollen wir sechzehn 8zackige Sterne ineinanderschachteln (siehe Abb. 2.24).

```
4460 REM ------------------------------
4470 REM ---  Testdaten fuer Stern  ---
4480 REM ------------------------------
4490 :
4500 DEG
4510 :
4520 radiusx1 = 100
4530 radiusx2 = 200
4540 radiusy1 = 50
4550 radiusy2 = 100
4560 kanf     = 0
4570 kend     = 360
4580 kstep    = 12
4590 :
4600 GOSUB 57000
4610 :
4620 GOSUB 40000
4630 :
4640 kstep    = 22.5 : REM nur gerade Teiler
4650 :
4660 FOR i=20 TO 170 STEP 10
4670    radiusx1 = i
4680    radiusx2 = i*2
4690    radiusy1 = i/1.2
4700    radiusy2 = i*1.2
4710    GOSUB 57000
4720 NEXT
4730 :
4740 GOSUB 40000
4750 :
```

Programm 2.22: Beispiele für die Anwendung des Stern-Unterprogramms

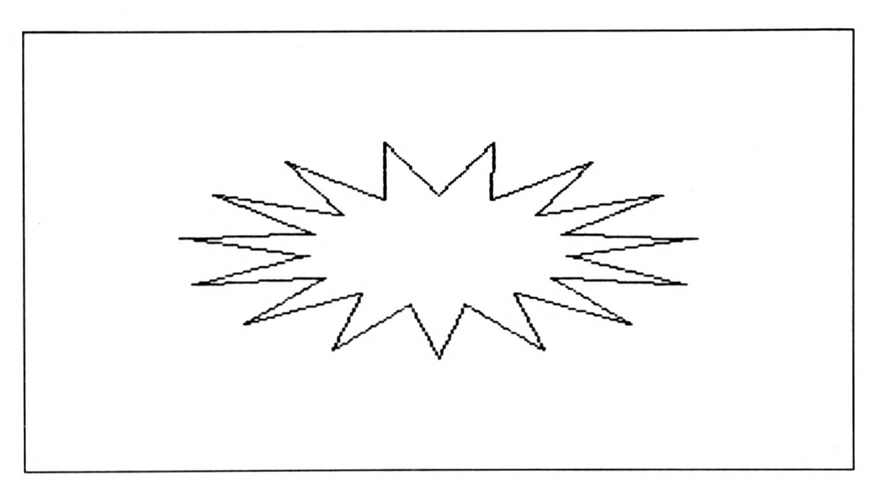


Abb. 2.23: Ein 15zackiger Stern, Schrittweite 12

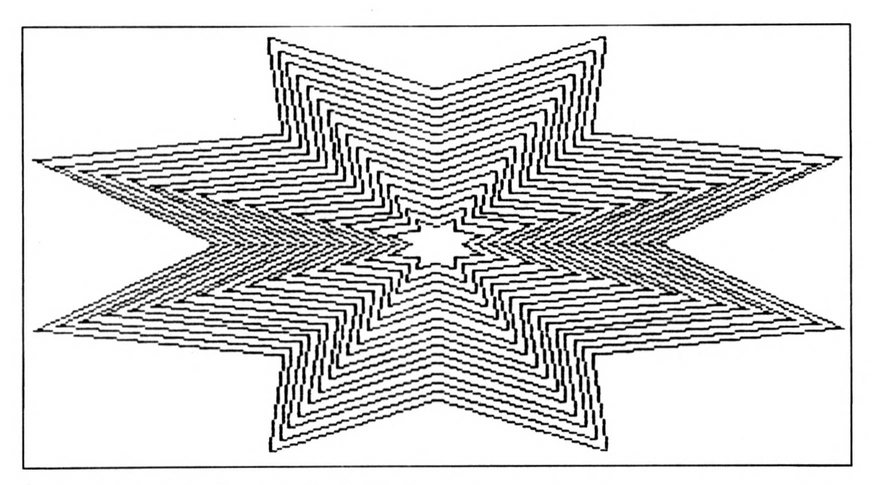


Abb. 2.24: Ineinandergeschachtelte Sterne

Dabei werden durch die Programmschleife ab Zeile 4660 jeweils die Radien um einen geringen Betrag erhöht.

Andere Möglichkeiten für Rechteck und Block

Bisher haben wir ein Unterprogramm zur Darstellung eines Rechtecks anhand von zwei gegebenen Eckpunkten auf dem Bildschirm benutzt. Dies ist wohl auch die gebräuchlichste Art der Parameterangabe. Aber auch noch ein ande-

```
58000 REM ----------------------------
58010 REM --- Rechteck mit Laengen  ---
58020 REM ----------------------------
58030 :
58040    MOVE rechtx,rechty
58050    DRAWR laenge,0,farbe
58060    DRAWR 0,-hoehe,farbe
58070    DRAWR -laenge,0,farbe
58080    DRAWR 0,hoehe,farbe
58090 :
58100 RETURN
58110 :
```

Programm 2.23: Unterprogramm zum Zeichnen eines Rechtecks (benötigt Längenangaben als Parameter)

rer Anwendungsfall ist denkbar: daß z. B. ein Bezugspunkt (Ecke oben links) gegeben ist und nur die Höhe und die Breite des Rechtecks. Diesem Umstand trägt das in Programm 2.23 gezeigte Unterprogramm Rechnung.

Zunächst wird der Grafik-Cursor in die bekannte linke obere Ecke des Rechtecks bewegt, und anschließend werden mit dem DRAWR-Befehl die einzelnen Kanten gezogen. Dabei bleibt jeweils eine Koordinate unverändert, zunächst die Y-Koordinate, so daß die obere Linie gezogen wird. Dann wird auf der rechten Seite des Rechtecks die Linie nach unten gezogen, daher das negative Vorzeichen. Bei der dritten Linie bleibt wiederum die Y-Koordinate unverändert, und mit negativem Vorzeichen wird die Linie von der rechten unteren Ecke zur linken unteren Ecke gezogen. Abschließend wird das Rechteck an der linken Seite geschlossen.

```
4760 REM -----------------------------
4770 REM ---  Testdaten fuer       ---
4780 REM ---   Rechteck mit Laengen ---
4790 REM -----------------------------
4800 :
4900 farbe = 2
4910 :
4920 FOR i=0 TO 700
4930    rechtx = i
4940    rechty = 200+150*COS(i)
4950    laenge = i/10
4960    hoehe  = i/15
4970    GOSUB 58000
4980 NEXT
4990 :
5000 GOSUB 40000
5010 :
```

Programm 2.24: Beispielanwendung des neuen Rechteck-Unterprogramms

Bitte achten Sie darauf, daß Sie im Hauptprogramm an anderer Stelle die Variablen laenge und hoehe nicht verwenden.

Beispiele dazu zeigen die Abbildungen 2.25 und 2.26.

Zunächst sehen Sie die Ausgabe ohne Zeile 4905 (siehe Abb. 2.25).

An diesem Beispiel wird der Unterschied zwischen der Berechnung im Gradmaß und im Bogenmaß bei gleichen Parametern sofort deutlich. Unterstützt

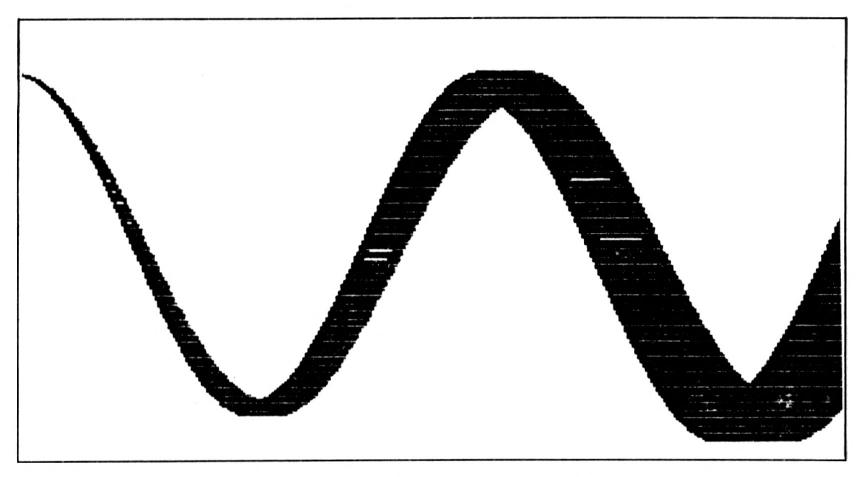


Abb. 2.25: Bildschirmausgabe des Programms 2.24 (ohne Zeile 4905)

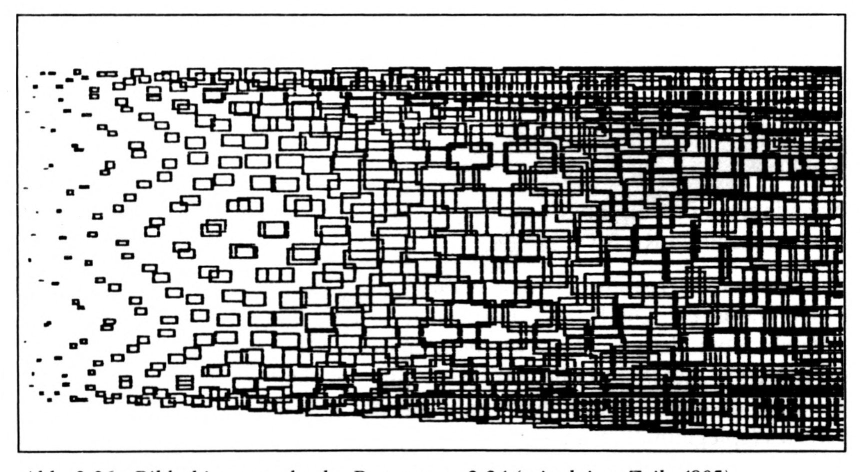


Abb. 2.26: Bildschirmausgabe des Programms 2.24 (mit aktiver Zeile 4905)

```
59000 REM ----------------------------
59010 REM ---   Block mit Laengen   ---
59020 REM ----------------------------
59030 :
59040    FOR block=0 TO laenge-1
59050       MOVE rechtx+block,rechty
59060       DRAWR 0,hoehe,farbe
59070    NEXT
59080 :
59090 RETURN
59100 :
```

Programm 2.25: Unterprogramm zum Zeichnen von Blöcken (benötigt Längenangaben als Parameter)

wird das Ganze durch die vom Rechner automatisch aufgerufene Modulo-Funktion. In Abb. 2.26 finden Sie die Hardcopy-Ausgabe mit aktiver Zeile 4905.

Welcher Parameter was bewirkt, sollten Sie durch Verändern der Parameter ausprobieren.

Abschließend noch das Unterprogramm zum Darstellen von Blöcken mittels Längenangaben, wobei Sie bitte die Testdaten ab Zeile 1390 verwenden wollen (Programm 2.25).

Die Programmschleife ist hier natürlich koordinatenunabhängig, da eine Längenangabe vorliegt. Eine weitere Möglichkeit wäre die Darstellung des Blocks mit waagerechten Linien, wobei dies sogar den Vorzug hätte, daß man die Schrittweite 2 verwenden kann, da ja jeweils zwei Y-Koordinaten durch einen Bildschirmpunkt dargestellt werden. Sicherlich eine gute Übung mit den bisherigen Erkenntnissen.

Vollkreis

Nachdem wir das ausgefüllte Rechteck vorgestellt haben, wollen wir nun auch den ausgefüllten Kreis vorstellen.

Wie Sie aus dem Listing ersehen können, durchläuft die Programmschleife nur einen Halbkreis. Die Schrittweite wurde so gewählt, daß zwar ein Vollkreis entsteht, aber nicht zu viele Zwischenpunkte durchgerechnet werden müssen. Der DRAW-Befehl in Zeile 60060 zieht nun von dem errechneten Punkt eine Linie zu dem entsprechenden Punkt auf der gegenüberliegenden Seite des Kreises (spiegelbildlich an der Senkrechten).

Auch hier wollen wir wieder Testdaten vorstellen (Programm 2.27).

```
60000 REM -----------------------------
60010 REM ---     Vollkreis         ---
60020 REM -----------------------------
60030 :
60034 DEG    : REM ggf setzen
60036 :
60040     FOR klauf=0 TO 180 STEP 60/radiusy
60050         MOVE mittelx+radiusx*SIN(klauf),mittely+rad
iusy*COS(klauf)
60060         DRAW mittelx+radiusx*SIN(0-klauf),mittely+r
adiusy*COS(0-klauf),farbe
60070     NEXT
60080 :
60090 RETURN
60100 :
```

Programm 2.26: Unterprogramm zum Zeichnen eines Vollkreises

```
5020 REM ------------------------------
5030 REM ---Testdaten fuer Vollkreis---
5040 REM ------------------------------
5050 :
5060 FOR i=20 TO 300 STEP 40
5070    mittelx = i*1.6
5080    mittely = i
5090    radiusx = i/3
5100    radiusy = i/5
5110    REM kstep   = 90/i
5120    GOSUB 60000
5130 NEXT
5140 :
5150 GOSUB 40000
5160 :
```

Programm 2.27: Anwendung des Vollkreis-Unterprogramms

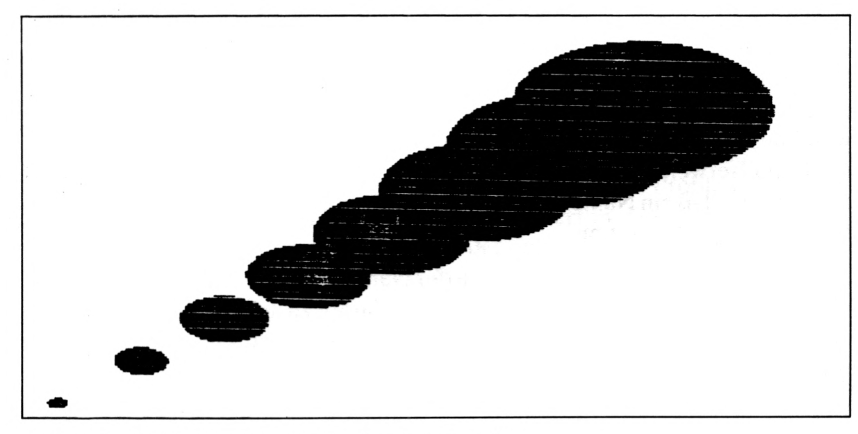


Abb. 2.27: Bildschirmausgabe des Programms 2.27

Wie aus Abb. 2.27 ersichtlich, werden von unten links ausgehend immer größer werdende Vollkreise gezeichnet, die wie an einer Schnur aufgereiht sind.

2.2 ERWEITERUNGEN ALS BEFEHLSERGÄNZUNG

Nachdem wir einige Ergänzungen in Form von BASIC-Unterprogrammen kennengelernt haben, wollen wir die wichtigsten Routinen – teilweise noch etwas verbessert – in Maschinensprache darstellen. Dabei soll zunächst Grundsätzliches zu Befehlserweiterungen behandelt werden.

Allgemeines zu BASIC-Erweiterungen

Die Befehlserweiterungen, von denen hier die Rede sein soll, sind eigentlich keine reinen BASIC-Befehle. Diese Befehle werden, wie etwa die Befehle des CP/M-ROM, in das Betriebssystem eingebunden. Sie können also vom BASIC, von Assemblerprogrammen oder von Pascal-Compilern – wenn sie diese Möglichkeit bieten – aufgerufen werden.

Diese Art von Befehlserweiterungen bestehen aus einem bis zu 16 Zeichen langen Namen. Für BASIC darf der Name nur aus Zeichen bestehen, die auch für Variablen erlaubt sind. Kleine Buchstaben sind nicht erlaubt, sie werden in Großbuchstaben umgewandelt. Erlaubt sind also Großbuchstaben, Zahlen und der Punkt. Weiterhin muß bei dem letzten Zeichen das höchstwertige Bit gesetzt sein, damit das Namensende erkannt wird. Bei Assemblerprogrammierung darf ein solcher Name jedoch aus beliebigen Zeichen bestehen. Dort gilt als einzige Einschränkung die Länge und die Namenendekennung.

Unter BASIC muß jedoch eine Befehlserweiterung mit einem bestimmten Symbol gekennzeichnet werden. Dieses Zeichen erreicht man durch Drücken einer SHIFT-Taste und der Taste rechts neben dem P, dem „Klammeraffen“: Man erhält einen senkrechten Strich. Dieses Zeichen wird im deutschen Zeichensatz als ö (Kleinbuchstabe) dargestellt. Es ist unter BASIC für die Anwendung einer Befehlserweiterung unerläßlich und muß vor jedem Namen einer Befehlserweiterung eingegeben werden. Nur durch dieses Symbol erkennt das BASIC, daß ein Name einer Befehlserweiterung folgt, und wandelt diesen Namen entsprechend um.

Allerdings unterliegt die Begrenzung der Namenslänge immer noch der Überwachung durch den Programmierer. Er hat darauf zu achten, daß der Befehlsname die Maximallänge von 16 Zeichen nicht übersteigt. Die Befehlserweiterung wird auch mit dem Kürzel RSX (Resident System eXtension) bezeichnet und ist damit nichts anderes als eine Systemerweiterung.

Im Prinzip repräsentiert die Befehlserweiterung unter BASIC nur einen

CALL-Befehl. Der Vorteil ist jedoch, daß man sich unter einem Befehlsnamen wesentlich mehr vorstellen kann als unter einer Adresse. Jetzt werden Sie sagen, man könne doch einen Variablennamen mit dieser Adresse belegen und hätte dann die gleiche Aussagekraft. An dieser Stelle muß entgegnet werden, daß sich die Einsprungadresse ändern kann und außerdem nach einem CLEAR, RUN oder dem LOAD bzw. MERGE eines BASIC-Programms die Variablen gelöscht werden. Springt man dann ein Maschinenprogramm mit dem CALL-Befehl an, so führt dies zum RESET, da ein CALL 0 erfolgt, wenn die Variable den Wert 0 enthält. Außerdem kann man mit einer Befehlserweiterung direkt in ein Erweiterungs-ROM springen.

Eine Befehlserweiterung steht nach dem Einschalten jedem CPC-Anwender zur Verfügung, es ist der Befehl |BASIC. Bevor Sie jedoch diesen Befehl ausprobieren, müssen Sie Ihr Programm speichern. Dieser Befehl initialisiert nämlich den BASIC-Interpreter neu.

Die Übergabe von Parametern erfolgt bei CALL-Befehlen und bei Befehlserweiterungen in der gleichen Weise. Diese Befehle können unter BASIC nur Integer-Werte als Parameter übernehmen. Die Parameter werden mit Kommata an den Befehl angehängt. Die Parameter können Werte zwischen −32768 und 32767 aufweisen. Werden Werte im Bereich von 32768 bis 65535, Fließkommawerte oder gar BASIC-Funktionen als Parameter angegeben, so werden die angegebenen Werte zuvor durch BASIC immer auf eine Integer-Zahl gebracht. Nur bei Werten, die kleiner −32768 oder größer als 65535 sind, verweigert BASIC mit einer Overflow-Fehlermeldung die Ausführung des Befehls.

Eine Verwendung als Funktion ist nicht ohne weiteres möglich, da der Befehlsname nicht als Argument einer BASIC-Zuweisung zugelassen ist. Solange sich die Befehlserweiterung auf die Verarbeitung von Integer-Werten beschränkt und nur eine Auswirkung auf den Bildschirm, den Drucker, das Sound-IC oder den Systembus hat, wird aber nicht mehr benötigt.

Was aber, wenn Fließkommawerte benötigt werden oder ein numerisches bzw. String-Ergebnis zurückgeliefert werden soll? Dieses Problem bekommt man durch die Verwendung des Variablen-Pointers in den Griff. Der Variablen-Pointer liefert nämlich die Adresse der angegebenen Variablen zurück, und diese ist zwangsläufig ein Integer-Wert.

Der Variablen-Pointer ist im CPC-BASIC das kaufmännische „At“. Sie werden ihn wahrscheinlich eher unter der Bezeichnung „Klammeraffe“ kennen. Im deutschen Zeichensatz wird er als Paragraph geführt. Stellt man dieses Symbol vor eine – bereits definierte – Variable, so liefert dies die Adresse dieser Variablen zurück. Bei Integer- und Fließkommazahlen zeigt diese

Adresse dann direkt auf das erste Byte des Wertes. Bei String-Variablen zeigt sie auf das erste Byte eines sogenannten Stringdeskriptors. Dies ist eine Folge von drei Bytes. Das erste Byte gibt die Länge und die beiden nächsten die Adresse des Strings an.

Bliebe nur noch zu sagen, daß maximal 32 Parameter übergeben werden dürfen. Die Parameterzahl ist begrenzt, damit der Systemstapel des Mikroprozessors nicht Teile des Betriebssystems überschreibt.

Erzeugung einer Befehlserweiterung

Befehlserweiterungen lassen sich grundsätzlich nur durch Maschinenprogramme erzeugen, eine Zusammenfassung mehrerer BASIC-Befehle zu einem neuen Befehl ist nicht möglich. Zur Erzeugung von RSX-Kommandos ist im wesentlichen nur eine Sprungtabelle, eine Namenstabelle und ein Sprung ins Betriebssystem nötig. In der Sprungtabelle müssen mindestens genauso viele Sprünge stehen, wie in der Namenstabelle an Namen vorhanden sind.

Weiterhin muß dem Betriebssystem noch ein vier Byte langer freier Speicherbereich je Befehlstabelle angegeben werden. Dieser wird dazu benutzt, eine Kette von einem Eintrag zum nächsten aufzubauen. Dazu wird beim neuesten Eintrag immer der davor geschaffene eingetragen. Der neueste wird immer in ein paar Systemspeicherzellen untergebracht. So wird bei der Suche nach einer RSX die Reihe der Einträge immer vom neuesten zum ältesten untersucht.

Mit dieser Vorgehensweise können sogar die Auswirkungen alter Einträge unterbunden werden, indem eine neue RSX mit dem gleichen Namen generiert wird. Diese Speicherung der Einträge hat aber auch Nachteile. Wird nämlich ein Eintrag zweimal oder öfter initialisiert, steht in dem Eintrag, daß der nächste er selbst ist. Das Betriebssystem sucht also ständig im Kreis, wenn ein älterer Eintrag gesucht wird. In diesem Fall ist die einzige Hilfe ein RESET.

Wird aber der zuletzt zur Verfügung gestellte Bereich gelöscht, so findet das Betriebssystem nur den letzten Eintrag, und das BASIC bringt bei allen älteren Befehlen die Meldung „unknown command". Damit reagiert das Betriebssystem wie bei nicht initialisierten Befehlen.

Die Initialisierung geschieht mit einem Sprung zur Adresse &BCD1, wobei im HL-Register die Adresse des vier Byte langen freien Speicherbereichs und im BC-Register die Adresse der Sprungtabelle übergeben werden muß.

Die ersten zwei Bytes der Sprungtabelle müssen die Adresse des Anfangs der entsprechenden Namenstabelle darstellen. Sowohl die vier freien Bytes als auch die Tabelle mit den RSX-Kommandos müssen im Speicherbereich von &4000 bis &BFFF liegen, da sonst Konflikte mit den ROMs auftreten können.

Ganz abgesehen davon ist die Belegung des unteren BASIC-Speichers oder des Bildschirmspeichers mit Befehlserweiterungen nicht sinnvoll. Die Tabelle der Befehlsnamen muß durch ein Byte mit dem Wert 0 abgeschlossen werden.

Die neuen Grafikbefehle – Übersicht

Insgesamt wurde der Befehlssatz um 13 Grafikbefehle erweitert, wovon allerdings drei Befehle die gleichen Auswirkungen haben. Die Grafikbefehle verändern die Lage des Koordinatenursprungs und des Grafik-Cursors nicht. Bis auf den Befehl |GRPEN wird auch die Farbe des Grafikstifts nicht verändert. Ebenso ist |GRPAPER der einzige Befehl, der die Grafikpaperfarbe verändert. Die Befehle |GRPEN bzw. |GRPAPER entsprechen den CPC 664- und CPC 6128-Befehlen GRAPHICS PEN bzw. GRAPHICS PAPER und werden daher auf diesen Rechnern *nicht* initialisert. Die Einstellung von DEG oder RAD wird nur innerhalb der Grafik-Routinen geändert. Die entsprechende Speicherzelle wird so zurückgelassen, wie sie vorgefunden wurde.

Die Befehle berücksichtigen die Lage des Grafikursprungs, lediglich in der Nähe der Extremwerte für den ORIGIN ist mit unvorhersehbaren Ergebnissen zu rechnen, da die Summe von ORIGIN und Koordinaten im Integer-Bereich liegen muß.

Achtung: Bei allen Befehlen müssen alle Parameter angegeben werden; geschieht dies nicht, so wird der Befehl nicht ausgeführt.

Es folgt eine Kurzbeschreibung der Befehle.

|RECHTECK,x1,y1,x2,y2,farbe

Dieser Befehl bewirkt das Zeichnen eines Rechtecks in der angegebenen Farbe. Dabei ist es egal, ob zuerst die größeren oder die kleineren Koordinatenwerte angegeben werden. Die Ecken des Rechtecks werden durch die Koordinaten (x1,y1), (x1,y2), (x2,y2) und (x2,y1) bestimmt. Da die Lage des Grafikursprungs berücksichtigt wird, sind, wenn der ORIGIN z. B. in der Mitte liegt, auch negative Koordinaten mit entsprechender Wirkung zugelassen.

|BLOCK,x1,y1,x2,y2,farbe

Der zweite Befehl bewirkt die Zeichnung eines ausgefüllten Rechtecks. Wie bei |RECHTECK ist die Reihenfolge der größeren und kleineren Koordinaten frei wählbar. Ebenso erfolgt die Berücksichtigung des Grafikursprungs. Die Eckpunkte ergeben sich analog zum Rechteck.

|*KREIS*,mittex,mittey,radiusx,radiusy,farbe

Das ist ein Befehl für eine(n) komplette(n) Ellipse/Kreis um den Mittelpunkt (mittex,mittey). Die Radien in X- und Y-Richtung können unterschiedlich gewählt werden, dadurch ist das Zeichnen von Ellipsen möglich. Die Ellipse wird in der angegebenen Farbe gezeichnet, und der ORIGIN wird dabei berücksichtigt.

|*SCHEIBE*,mittex,mittey,radiusx,radiusy,farbe

Dieser Befehl wirkt ähnlich wie |KREIS, nur daß der Kreis/die Ellipse in der angegebenen Farbe *ausgefüllt* gezeichnet wird. Der Mittelpunkt wird ebenfalls unter Berücksichtigung des ORIGIN gesetzt.

|*BOGEN*,mittex,mittey,radiusx,radiusy,wianf,wiend,wistep,farbe

Der fünfte Befehl zeichnet den Bogen einer Ellipse um den angegebenen Mittelpunkt. Dabei wird beim Winkel wianf angefangen, um jeweils wistep erhöht und, wenn wiend erreicht ist, wieder aufgehört. Die Winkel werden im Gradmaß angegeben und können auch negativ sein.

|*BOGEN.D*,mittex,mittey,radiusx,radiusy,@wianf,@wiend, @wistep,farbe

Dieser Befehl hat die gleichen Auswirkungen wie der vorige. Der einzige Unterschied liegt darin, daß die im Gradmaß angegebenen Winkel zuvor in Variablen abgelegt werden müssen. Die Werte werden dann mit dem Variablen-Pointer übergeben; das hat den Vorteil, daß für die Winkel auch Gradbruchteile erlaubt sind. Ebenso können die übergebenen Winkel den vollen Bereich von Fließkommazahlen ausnutzen.

|*BOGEN.R*,mittex,mittey,radiusx,radiusy,@wianf,@wiend, @wistep,farbe

Dieser Befehl wird genauso benutzt wie |BOGEN.D, mit dem Unterschied, daß in den zu übergebenden Variablen die Winkel nicht im Gradmaß, sondern im Bogenmaß angegeben werden.

|*RADIUS*,mittex,mittey,radiusx,radiusy,winkel,farbe

Dieser Befehl zeichnet einen Radius in einer (imaginären) Ellipse unter dem angegebenen Winkel ein. Der Winkel wird im Gradmaß angegeben und kann nur ganzzahlige Werte annehmen.

|*GRPEN,farbe*

Dieser Befehl setzt den Grafikstift auf die angegebene Farbe. Er kann benutzt werde, um z. B. die Farbe bei der Grafikausgabe von Zeichen umzuschalten. Dies wäre unter BASIC nur mit einem PLOT oder DRAW auf eine nicht im Grafikfenster liegende Grafikposition möglich. |GRPEN ist nur für den CPC 464 verfügbar, da die neueren Schneider Computer den Befehl GRAPHICS PEN besitzen.

|*GRPAPER,farbe*

Der Befehl |GRPAPER setzt den Grafikhintergrund auf die angegebene Farbe. Unter BASIC ließe sich die Grafikpapierfarbe sonst nur mittels eines „CLG farbe" umschalten, wobei der Inhalt des Grafikfensters gelöscht würde. Dieser Befehl ist nur auf dem CPC 464 ansprechbar, da für die anderen CPCs der Befehl GRAPHICS PAPER vorhanden ist.

|*GRMODE,modus*

Dieser Befehl schaltet den Ausgabemodus für die Grafik um. Normalerweise wird dies unter BASIC mit der folgenden Befehlssequenz bewirkt:

```
PRINT CHR$(23) ;CHR$ (modus) ;
```

Der Parameter modus kann Werte zwischen 0 und 3 annehmen (vgl. Kapitel 5).

0 ist der normale überschreibende,
1 der EXCLUSIV-ODER (XOR) verknüpfende,
2 der UND verknüpfende und schließlich
3 der ODER verknüpfende Grafikmodus.

Beim CPC 664 und CPC 6128 kann der Verknüpfungsmodus auch als vierter Parameter bei den Befehlen DRAW, DRAWR, MOVE, MOVER, PLOT und PLOTR angegeben werden.

|*QUADER,x1,y1,x2,y2,linkshintenx,untenhinteny,farbe*

Der Befehl zeichnet einen Quader in der angegebenen Farbe. Der Quader wird als Drahtmodell dargestellt, es sind also nur die Kanten sichtbar. Die ersten vier Koordinaten geben die Vorderfläche des Quaders an. Das Koordinatenpaar linkshintenx und untenhinteny gibt die untere linke hintere Ecke des Quaders an. Wie üblich wird die Lage des Ursprungs berücksichtigt.

|VOLLQUADER,x1,y1,x2,y2,linkshx,untenhy,farbe1,farbe2,farbe3

Der Befehl zeichnet einen Quader mit ausgefüllten Flächen. Die Parameter x1, y1, x2 und y2 beschreiben wieder die Vorderfläche, die Parameter linkshx und untenhy wieder die untere linke hintere Ecke des Quaders. Farbe1 gibt den Grafikstift für die Vorderfläche an, farbe2 den Stift für die Seitenfläche und farbe3 die Farbe des Bodens bzw. Deckels, je nachdem welche Flächen bei dem Quader zu sehen sind. Die Kanten der zusammenstoßenden Flächen werden mit dem Grafikstift 0 nachgezogen. Dieser Befehl kann sehr gut für Diagramme mit dreidimensionaler Sichtweise eingesetzt werden.

```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

00FF              10 JA:      EQU  #FF
0000              20 NEIN:    EQU  #0000
                  30
00FF              40 CPC464:  EQU  JA
0000              50 CPC664:  EQU  NEIN
0000              60 CPC128:  EQU  NEIN
                  70
BCD1              80 ERWEIT:  EQU  #BCD1
BBC0              90 MOVE:    EQU  #BBC0
BBC6             100 XY_POS:  EQU  #BBC6
BBC9             110 SETORG:  EQU  #BBC9
BBCC             120 GETORG:  EQU  #BBCC
BBDE             130 SGRPEN:  EQU  #BBDE
BBE1             140 GGRPEN:  EQU  #BBE1
BBE4             150 SGRPAP:  EQU  #BBE4
BBEA             160 PLOT:    EQU  #BBEA
BBF6             170 DRAW:    EQU  #BBF6
BBF9             180 DRAWR:   EQU  #BBF9
BC11             190 GTMODE:  EQU  #BC11
BC59             200 SGRMOD:  EQU  #BC59
                 210
                 220
8000             230          ORG  #8000
                 240
8000  010FB0     250 INIT:    LD   BC,TABELLE          ; Adr. d. Sprungtabelle
8003  218D80     260          LD   HL,EXTTAB           ; Adr. v. 4 Byte f. Sys.
8006  CDD1BC     270          CALL ERWEIT              ; Befehle einbinden
8009  3EC9       280          LD   A,#C9               ; A mit RET laden
800B  320080     290          LD   (INIT),A            ; mehrfaches Einbinden
800E  C9         300          RET                      ; verhindern
800F  3880       310 TABELL:  DEFW NAMEN               ; Adr. d. entspr. Namen
8011  C39180     320          JP   RE_ECK              ; zur Rechteckroutine
8014  C3D480     330          JP   BLOCK               ; zur Blockroutine
8017  C31D81     340          JP   KREIS               ; zum Kreis
801A  C32781     350          JP   SCHEIBE             ; zur Scheibe
801D  C31682     360          JP   BOGEN               ; zum Bogen (Gradmaß)
8020  C33A82     370          JP   BOGEND              ; zum Bogen (Gradmaß)
8023  C34282     380          JP   BOGENR              ; zum Bogen (Bogenmaß)
8026  C3CE82     390          JP   RADIUS              ; zum Radius (Gradmaß)
00FF             400          IF   CPC464              ; NUR für CPC 464
8029  C31383     410          JP   GRPEN               ; zum Grafikstiftsetzen
802C  C31C83     420          JP   GRPAP               ; zum Grafikpapersetzen
802F             430          END                      ; NICHT für CPC 664/6128
802F  C32583     440          JP   GRMODE              ; zum Grafikmodussetzen
8032  C32E83     450          JP   QUADER              ; zum (leeren) Quader
8035  C3C883     460          JP   VOLLQU              ; zum (vollen) Quader
8038  52454348   470 NAMEN:   DEFB "R","E","C","H","T","E","C","K"+#80
8040  424C4F43   480          DEFB "B","L","O","C","K"+#80
8045  4B524549   490          DEFB "K","R","E","I","S"+#80
804A  53434845   500          DEFB "S","C","H","E","I","B","E"+#80
8051  424F4745   510          DEFB "B","O","G","E","N"+#80
8056  424F4745   520          DEFB "B","O","G","E","N",".","D"+#80
805D  424F4745   530          DEFB "B","O","G","E","N",".","R"+#80
```

Programm 2.28: Das Assembler-Listing der Grafikerweiterung (ff.)

```
8064  52414449   540           DEFB "R","A","D","I","U","S"+#80
00FF             550           IF   CPC464          ; NUR für CPC 464
806A  47525045   560           DEFB "G","R","P","E","N"+#80
806F  47525041   570           DEFB "G","R","P","A","P","E","R"+#80
8076             580           END                  ; NICHT für CPC 664/6128
8076  47524D4F   590           DEFB "G","R","M","O","D","E"+#80
807C  51554144   600           DEFB "Q","U","A","D","E","R"+#80
8082  564F4C4C   610           DEFB "V","O","L","L","Q","U","A","D","E","R"+#80
808C  00         620           DEFB #00
808D             630 EXTTAB:   DEFS 4               ; für Betr.sys. reserv.
                 640
00FF             650           IF   CPC464          ; NUR für CPC 464
BD40             660 HL_FLO:   EQU  #BD40
BD46             670 FLO_HL:   EQU  #BD46
BD58             680 FLOADD:   EQU  #BD58
BD5E             690 FLOSB2:   EQU  #BD5E
BD61             700 FLOMUL:   EQU  #BD61
BD64             710 FLODIV:   EQU  #BD64
BD6A             720 FLOVGL:   EQU  #BD6A
BD6D             730 FLOVZW:   EQU  #BD6D
BD70             740 FLOSGN:   EQU  #BD70
BD79             750 FLOSQR:   EQU  #BD79
BD88             760 FLOSIN:   EQU  #BD88
BD8B             770 FLOCOS:   EQU  #BD8B
BDA3             780 INTABS:   EQU  #BDA3
BDA9             790 ISIGNB:   EQU  #BDA9
BDAC             800 INTADD:   EQU  #BDAC
BDAF             810 INTSB1:   EQU  #BDAF
BDB2             820 INTSB2:   EQU  #BDB2
BDC4             830 INTVGL:   EQU  #BDC4
BDC7             840 INTVZW:   EQU  #BDC7
BDCA             850 INTSGN:   EQU  #BDCA
B8F7             860 DEGREE:   EQU  #B8F7
8091             870           END                  ; Ende für CPC 464
                 880
0000             890           IF   CPC664          ; NUR für CPC 664
8091             900 HL_FLO:   EQU  #BD61
8091             910 FLO_HL:   EQU  #BD67
8091             920 FLOADD:   EQU  #BD79
8091             930 FLOSB2:   EQU  #BD7F
8091             940 FLOMUL:   EQU  #BD82
8091             950 FLODIV:   EQU  #BD85
8091             960 FLOVGL:   EQU  #BD8B
8091             970 FLOVZW:   EQU  #BD8E
8091             980 FLOSGN:   EQU  #BD91
8091             990 FLOSQR:   EQU  #BD9A
8091            1000 FLOSIN:   EQU  #BDA9
8091            1010 FLOCOS:   EQU  #BDAC
8091            1020 INTABS:   DEFB #D7,#2F,#1D,#C9
8091            1030 ISIGNB:   DEFB #D7,#3C,#1D,#C9
8091            1040 INTADD:   DEFB #D7,#4F,#1D,#C9
8091            1050 INTSB1:   DEFB #D7,#58,#1D,#C9
8091            1060 INTSB2:   DEFB #D7,#57,#1D,#C9
8091            1070 INTVGL:   DEFB #D7,#07,#DE,#C9
8091            1080 INTVZW:   DEFB #D7,#F2,#DD,#C9
8091            1090 INTSGN:   DEFB #D7,#FE,#DD,#C9
8091            1100 DEGREE:   EQU  #B113
8091            1110           END                  ; Ende für CPC 664
                1120
0000            1130           IF   CPC128          ; NUR für CPC 6128
8091            1140 HL_FLO:   EQU  #BD64
8091            1150 FLO_HL:   EQU  #BD6A
8091            1160 FLOADD:   EQU  #BD7C
8091            1170 FLOSB2:   EQU  #BD82
8091            1180 FLOMUL:   EQU  #BD85
8091            1190 FLODIV:   EQU  #BD88
8091            1200 FLOVGL:   EQU  #BD8E
8091            1210 FLOVZW:   EQU  #BD91
8091            1220 FLOSGN:   EQU  #BD94
8091            1230 FLOSQR:   EQU  #BD9D
8091            1240 FLOSIN:   EQU  #BDAC
8091            1250 FLOCOS:   EQU  #BDAF
8091            1260 INTABS:   DEFB #D7,#2A,#1D,#C9
8091            1270 ISIGNB:   DEFB #D7,#37,#1D,#C9
8091            1280 INTADD:   DEFB #D7,#4A,#1D,#C9
8091            1290 INTSB1:   DEFB #D7,#53,#1D,#C9
8091            1300 INTSB2:   DEFB #D7,#52,#1D,#C9
8091            1310 INTVGL:   DEFB #D7,#02,#1E,#C9
8091            1320 INTVZW:   DEFB #D7,#ED,#1D,#C9
```

```
8091            1330 INTSGN: DEFB #D7,#F9,#1D,#C9
8091            1340 DEGREE: EQU  #B113
8091            1350         END                    ; Ende für CPC 6128
                1360
                1370
8091 FE05       1380 RE_ECK: CP   5                 ; 5 Argumente?
8093 C0         1390         RET  NZ                ; wenn nicht, zum BASIC
8094 CD7285     1400         CALL GRSTO             ; Grafikwerte speichern
8097 CD1683     1410         CALL FARBE             ; Grafikstift setzen
809A CDC585     1420         CALL GTHL6X            ; y1 holen
809D 22BB80     1430         LD   (Y_INT),HL        ; y1 abspeichern
80A0 CDCC85     1440         CALL GTDE8X            ; x1 holen
80A3 ED53C380   1450         LD   (X_INT),DE        ; x1 abspeichern
80A7 CDB785     1460         CALL GTHL2X            ; y2 holen
80AA CDBE85     1470         CALL GTDE4X            ; x2 holen
80AD CDB380     1480         CALL R_ECK             ; Rechteck x1,y1,x2,y2
80B0 C38D85     1490         JP   GRREST            ; Grafik auf alte Werte
                1500
80B3 D5         1510 R_ECK:  PUSH DE                ; x2 retten
80B4 E5         1520         PUSH HL                ; y2 retten
80B5 D5         1530         PUSH DE                ; x2 nochmal retten
80B6 CDC0BB     1540         CALL MOVE              ; MOVE x2,y2
80B9 D1         1550         POP  DE                ; x2 holen
80BA 210000     1560         LD   HL,#0000          ; y1 laden
80BB            1570 Y_INT:  EQU  $-2               ; Argument v. LD HL,....
80BD E5         1580         PUSH HL                ; y1 retten
80BE CDF6BB     1590         CALL DRAW              ; DRAW x2,y1
80C1 E1         1600         POP  HL                ; y1 holen
80C2 110000     1610         LD   DE,#0000          ; x1 laden
80C3            1620 X_INT:  EQU  $-2               ; Argument v. LD DE,....
80C5 D5         1630         PUSH DE                ; x1 retten
80C6 CDF6BB     1640         CALL DRAW              ; DRAW x1,y1
80C9 D1         1650         POP  DE                ; x1 holen
80CA E1         1660         POP  HL                ; y2 holen
80CB E5         1670         PUSH HL                ; y2 retten
80CC CDF6BB     1680         CALL DRAW              ; DRAW x1,y2
80CF E1         1690         POP  HL                ; y2 holen
80D0 D1         1700         POP  DE                ; x2 holen
80D1 C3F6BB     1710         JP   DRAW              ; DRAW x2,y2 und Return
                1720
80D4 FE05       1730 BLOCK:  CP   5                 ; 5 Argumente?
80D6 C0         1740         RET  NZ                ; wenn nicht, zum BASIC
80D7 CD7285     1750         CALL GRSTO             ; Grafikwerte speichern
80DA CD1683     1760         CALL FARBE             ; Grafikstift setzen
80DD CDCC85     1770         CALL GTDE8X            ; x1 holen
80E0 ED53FF80   1780         LD   (X1),DE           ; x1 abspeichern
80E4 CDBE85     1790         CALL GTDE4X            ; x2 holen
80E7 ED530781   1800         LD   (X2),DE           ; x2 abspeichern
80EB CDB785     1810         CALL GTHL2X            ; y2 holen
80EE EB         1820         EX   DE,HL             ; y2 nach DE
80EF CDC585     1830         CALL GTHL6X            ; y1 holen
80F2 CDA485     1840         CALL VERGL             ; vergleichen
80F5 22BB80     1850         LD   (Y_INT),HL        ; obeny speichern
80F8 CB83       1860         RES  0,E               ; ==> unteny gerade
80FA 2ABB80     1870 ZEILE:  LD   HL,(Y_INT)        ; y holen
80FD D5         1880         PUSH DE                ; unteny retten
80FE 110000     1890         LD   DE,#0000          ; x1 laden
80FF            1900 X1:     EQU  $-2               ; Argument v. LD DE,....
8101 E5         1910         PUSH HL                ; y retten
8102 CDC0BB     1920         CALL MOVE              ; MOVE x1,y
8105 E1         1930         POP  HL                ; y holen
8106 110000     1940         LD   DE,#0000          ; x2 laden
8107            1950 X2:     EQU  $-2               ; Argument v. LD DE,....
8109 E5         1960         PUSH HL                ; y retten
810A CDF6BB     1970         CALL DRAW              ; DRAW x2,y
810D E1         1980         POP  HL                ; y holen
810E 2B         1990         DEC  HL                ; y = y - 2
810F 2B         2000         DEC  HL
8110 22BB80     2010         LD   (Y_INT),HL        ; y speichern
8113 D1         2020         POP  DE                ; unteny holen
8114 B7         2030         OR   A                 ; Carry löschen
8115 ED52       2040         SBC  HL,DE             ; unteny < y?
8117 F2FA80     2050         JP   P,ZEILE           ; wenn ja, nächste Linie
811A C38D85     2060         JP   GRREST            ; Grafik auf alte Werte
                2070
811D FE05       2080 KREIS:  CP   5                 ; 5 Argumente?
811F C0         2090         RET  NZ                ; wenn nicht, zum BASIC
8120 3EFF       2100         LD   A,JA              ; Flag für leeren
8122 326D85     2110         LD   (FLAG),A          ; Kreis laden
```

```
8125  1807      2120          JR    WEITER        ; dort weitermachen
                2130
8127  FE05      2140 SCHEIB:  CP    5             ; 5 Argumente?
8129  C0        2150          RET   NZ            ; wenn nicht, zum BASIC
812A  AF        2160          XOR   A             ; Flag für vollen
812B  326D85    2170          LD    (FLAG),A      ; Kreis laden
812E  CD7285    2180 WEITER:  CALL  GRSTO         ; Grafikwerte speichern
8131  CD1683    2190          CALL  FARBE         ; Grafikfarbe setzen
8134  CDC585    2200          CALL  GTHL6X        ; mittey und
8137  CDCC85    2210          CALL  GTDE8X        ; mittex holen
813A  CD4685    2220          CALL  MITTE         ; Mittelpunkt setzen
813D  210000    2230          LD    HL,0          ; HL mit 0 laden und
8140  22ED84    2240          LD    (ALTX),HL     ; altx auf 0 setzen
8143  DDE5      2250          PUSH  IX            ; IX wegen Param. retten
8145  CDBE85    2260          CALL  GTDE4X        ; radiusx holen
8148  21F285    2270          LD    HL,RADX       ; Adr. v. radiusx^2
814B  EB        2280          EX    DE,HL         ; Adr. und Wert wechseln
814C  CDF584    2290          CALL  QUADRI        ; (DE) = Quadrat v. HL
814F  DDE1      2300          POP   IX            ; IX für Param. holen
8151  CDB785    2310          CALL  GTHL2X        ; radiusy holen
8154  E5        2320          PUSH  HL            ; radiusy retten
8155  11F785    2330          LD    DE,RADY       ; Adr. v. radiusy^2
8158  CDF584    2340          CALL  QUADRI        ; (DE) = Quadrat v. HL
815B  E1        2350          POP   HL            ; radiusy holen
815C  CDA3BD    2360          CALL  INTABS        ; Betrag von HL bilden
815F  CB85      2370          RES   0,L           ; gerade Koordinate!
8161  22F084    2380          LD    (ALTY),HL     ; Anfangswert f. alty
8164  22BB80    2390 Y_LOOP:  LD    (Y_INT),HL    ; Y speichern
8167  11ED85    2400          LD    DE,Y^2        ; Zwischensp. v. y^2
816A  CDF584    2410          CALL  QUADRI        ; HL zu y^2 wandeln
816D  11F785    2420          LD    DE,RADY       ; Adr. v. radiusy^2
8170  CD64BD    2430          CALL  FLODIV        ; y^2 / radiusy^2
8173  11E885    2440          LD    DE,EINS       ; Adr. v. Wert 1
8176  CD5EBD    2450          CALL  FLOSB2        ; 1 - y^2 / radiusy^2
8179  11F285    2460          LD    DE,RADX       ; Adr. v. radiusx^2
817C  CD61BD    2470          CALL  FLOMUL        ; radiusx^2*letztem Wert
817F  CD79BD    2480          CALL  FLOSQR        ; davon Quadratwurzel
8182  CD46BD    2490          CALL  FLO_HL        ; nach Integer
8185  22C380    2500          LD    (X_INT),HL    ; und speichern
8188  CD9781    2510          CALL  ZEICHNEN      ; damit zeichnen
818B  2ABB80    2520          LD    HL,(Y_INT)    ; Y-Koordinate holen
818E  7C        2530          LD    A,H           ; HL auf HL=0
818F  B5        2540          OR    L             ; prüfen und mit
8190  2B        2550          DEC   HL            ; y = y - 2
8191  2B        2560          DEC   HL            ; wenn HL<>0 war
8192  20D0      2570          JR    NZ,Y_LOOP     ; weiter, sonst beenden
8194  C38D85    2580          JP    GRREST        ; Grafik auf alte Werte
                2590
8197  2ABB80    2600 ZEICHN:  LD    HL,(Y_INT)    ; Y-Koordinate holen
819A  ED5BC380  2610          LD    DE,(X_INT)    ; X-Koordinate holen
819E  E5        2620          PUSH  HL            ; Y retten
819F  D5        2630          PUSH  DE            ; X retten
81A0  CDEABB    2640          CALL  PLOT          ; Grafikcursor setzen
81A3  3A6D85    2650          LD    A,(FLAG)      ; Flag holen
81A6  B7        2660          OR    A             ; und testen
81A7  2815      2670          JR    Z,VOLL1       ; wenn für voll, dorthin
81A9  2AF084    2680          LD    HL,(ALTY)     ; vorheriger Y-Wert
81AC  ED5BED84  2690          LD    DE,(ALTX)     ; vorheriger X-Wert
81B0  E5        2700          PUSH  HL            ; alty retten
81B1  D5        2710          PUSH  DE            ; altx retten
81B2  CDF6BB    2720          CALL  DRAW          ; Linie ziehen
81B5  E1        2730          POP   HL            ; altx holen
81B6  CDC7BD    2740          CALL  INTVZW        ; altx = -altx
81B9  D1        2750          POP   DE            ; alty holen
81BA  EB        2760          EX    DE,HL         ; -altx <--> alty
81BB  CDEABB    2770          CALL  PLOT          ; PLOT -altx,alty
81BE  E1        2780 VOLL1:   POP   HL            ; x holen
81BF  CDC7BD    2790          CALL  INTVZW        ; x = -x
81C2  D1        2800          POP   DE            ; y holen
81C3  EB        2810          EX    DE,HL         ; x zu DE und y zu HL
81C4  CDF6BB    2820          CALL  DRAW          ; DRAW -x,y
81C7  2ABB80    2830          LD    HL,(Y_INT)    ; y holen
81CA  CDC7BD    2840          CALL  INTVZW        ; y = -y
81CD  ED5BC380  2850          LD    DE,(X_INT)    ; x holen
81D1  E5        2860          PUSH  HL            ; -y retten
81D2  D5        2870          PUSH  DE            ; x retten
81D3  CDEABB    2880          CALL  PLOT          ; PLOT x,-y
81D6  3A6D85    2890          LD    A,(FLAG)      ; Flag holen
81D9  B7        2900          OR    A             ; ist es 0?
```

```
81DA  281A      2910           JR   Z,VOLL2       ; dann zu VOLL2
81DC  2AF084    2920           LD   HL,(ALTY)     ; sonst alty holen
81DF  CDC7BD    2930           CALL INTVZW        ; alty = -alty
81E2  ED5BED84  2940           LD   DE,(ALTX)     ; altx holen
81E6  E5        2950           PUSH HL            ; -alty retten
81E7  D5        2960           PUSH DE            ; altx retten
81E8  CDF6BB    2970           CALL DRAW          ; DRAW altx,-alty
81EB  E1        2980           POP  HL            ; altx holen
81EC  CDC7BD    2990           CALL INTVZW        ; altx = -altx
81EF  D1        3000           POP  DE            ; -alty holen
81F0  EB        3010           EX   DE,HL         ; -altx <--> -alty
81F1  CDEABB    3020           CALL PLOT          ; PLOT -altx,-alty
81F4  180A      3030           JR   K_LEER        ; weiter für Kreislinie
81F6  2ABB80    3040 VOLL2:    LD   HL,(Y_INT)    ; y holen
81F9  7C        3050           LD   A,H           ; überprüfen ob
81FA  B5        3060           OR   L             ; y <> 0?
81FB  2003      3070           JR   NZ,K_LEER     ; dann zeichnen
81FD  E1        3080           POP  HL            ; Stack für Austritt
81FE  D1        3090           POP  DE            ; setzen und
81FF  C9        3100           RET                ; Rückkehr
8200  E1        3110 K_LEER:   POP  HL            ; x holen
8201  CDC7BD    3120           CALL INTVZW        ; x = -x
8204  D1        3130           POP  DE            ; -y holen
8205  EB        3140           EX   DE,HL         ; -x <--> -y
8206  CDF6BB    3150           CALL DRAW          ; DRAW -x,-y
8209  2ABB80    3160           LD   HL,(Y_INT)    ; y holen
820C  22F084    3170           LD   (ALTY),HL     ; als alty speichern
820F  2AC380    3180           LD   HL,(X_INT)    ; x holen
8212  22ED84    3190           LD   (ALTX),HL     ; als altx speichern
8215  C9        3200           RET                ; und Rückkehr
                3210
8216  FE08      3220 BOGEN:    CP   8             ; 8 Argumente?
8218  C0        3230           RET  NZ            ; wenn nicht, zum BASIC
8219  CDB785    3240           CALL GTHL2X        ; wistep holen
821C  110B86    3250           LD   DE,WISTEP     ; Adr. v. wistep
821F  CD3485    3260           CALL INTFLO        ; HL zu (DE) wandeln
8222  CDBE85    3270           CALL GTDE4X        ; wiend holen
8225  210686    3280           LD   HL,WIEND      ; Adr. v. wiend
8228  EB        3290           EX   DE,HL         ; Adr. <--> Wert
8229  CD3485    3300           CALL INTFLO        ; HL zu wiend wandeln
822C  CDC585    3310           CALL GTHL6X        ; wianf holen
822F  11FC85    3320           LD   DE,WINKEL     ; Adr. v. wianf
8232  CD3485    3330           CALL INTFLO        ; HL zu wianf wandeln
8235  CD6085    3340           CALL DEG           ; Gradmaß setzen
8238  182A      3350           JR   BOCONT        ; zum Weitermachen
                3360
823A  FE08      3370 BOGEND:   CP   8             ; 8 Argumente?
823C  C0        3380           RET  NZ            ; wenn nicht, zum BASIC
823D  CD6085    3390           CALL DEG           ; Gradmaß setzen
8240  1806      3400           JR   BDCONT        ; zum Weitermachen
                3410
8242  FE08      3420 BOGENR:   CP   8             ; 8 Argumente?
8244  C0        3430           RET  NZ            ; wenn nicht, zum BASIC
8245  CD5585    3440           CALL RAD           ; Bogenmaß setzen
8248  CDB785    3450 BDCONT:   CALL GTHL2X        ; Adr. v. Var. wistep
824B  110B86    3460           LD   DE,WISTEP     ; Adr. v. wistep
824E  CDAB85    3470           CALL COPYHD        ; BASIC-Var. copieren
8251  CDBE85    3480           CALL GTDE4X        ; Adr. v. Var. wiend
8254  210686    3490           LD   HL,WIEND      ; Adr. v. wiend
8257  EB        3500           EX   DE,HL         ; Adr. vertauschen
8258  CDAB85    3510           CALL COPYHD        ; BASIC-Var. copieren
825B  CDC585    3520           CALL GTHL6X        ; Adr. v. Var. wianf
825E  11FC85    3530           LD   DE,WINKEL     ; Adr. v. wianf
8261  CDAB85    3540           CALL COPYHD        ; BASIC-Var. copieren
8264  CD7285    3550 BOCONT:   CALL GRSTO         ; Grafikwerte speichern
8267  CD1683    3560           CALL FARBE         ; Grafikpen einstellen
826A  CDE185    3570           CALL GTH14X        ; mittex holen
826D  CDDA85    3580           CALL GTD12X        ; mittey holen
8270  EB        3590           EX   DE,HL         ; vertauschen
8271  CD4685    3600           CALL MITTE         ; Mittelpunkt setzen
8274  CDCC85    3610           CALL GTDE8X        ; radiusy holen
8277  21F785    3620           LD   HL,RADY       ; Adr. von RADY
827A  EB        3630           EX   DE,HL         ; Adr. u. Wert tauschen
827B  CD3485    3640           CALL INTFLO        ; HL zu Float wandeln
827E  CDD385    3650           CALL GTH10X        ; radiusx holen
8281  11F285    3660           LD   DE,RADX       ; Adr. v. RADX
8284  CD3485    3670           CALL INTFLO        ; HL zu Float wandeln
8287  210B86    3680           LD   HL,WISTEP     ; Adr. v. wistep
828A  CD70BD    3690           CALL FLOSGN        ; wistep <= null?
```

```
828D  2839        3700          JR   Z,BOENDE       ; für wistep <= null
828F  3837        3710          JR   C,BOENDE       ; zum BASIC zurück!
8291  3EFF        3720          LD   A,JA           ; Flag für ersten Punkt
8293  32A582      3730          LD   (ERSTER),A     ; laden
8296  21FC85      3740          LD   HL,WINKEL      ; Adr. v. wianf
8299  E5          3750 WILOOP:  PUSH HL             ; Adr. v. winkel retten
829A  CDFD84      3760          CALL RECHNE         ; x und y berechnen
829D  ED5BC380    3770          LD   DE,(X_INT)     ; DE = x
82A1  2ABB80      3780          LD   HL,(Y_INT)     ; HL = y
82A4  3EFF        3790          LD   A,#FF          ; Flag für ersten Lauf
82A5              3800 ERSTER:  EQU  $-1            ; Argument v. LD A,..
82A6  B7          3810          OR   A              ; laden und prüfen
82A7  2809        3820          JR   Z,WEITE1       ; wenn 0, dort weiter
82A9  CDC0BB      3830          CALL MOVE           ; MOVE x,y
82AC  AF          3840          XOR  A              ; A löschen und
82AD  32A582      3850          LD   (ERSTER),A     ; Flag löschen
82B0  1803        3860          JR   WEITE2         ; dort weiter
82B2  CDF6BB      3870 WEITE1:  CALL DRAW           ; DRAW x,y
82B5  21FC85      3880 WEITE2:  LD   HL,WINKEL      ; Adr. v. winkel
82B8  110B86      3890          LD   DE,WISTEP      ; Adr. v. wistep
82BB  CD58BD      3900          CALL FLOADD         ; winkel=winkel+wistep
82BE  110686      3910          LD   DE,WIEND       ; Adr. v. wiend
82C1  EB          3920          EX   DE,HL          ; vertau. weg. Bedingung
82C2  CD6ABD      3930          CALL FLOVGL         ; vergleichen
82C5  E1          3940          POP  HL             ; Adr. v. winkel holen
82C6  30D1        3950          JR   NC,WILOOP      ; end>=lauf, dann weiter
82C8  CD8D85      3960 BOENDE:  CALL GRREST         ; Grafik auf alte Werte
82CB  C36C85      3970          JP   DEGRES         ; DEG/RAD herstellen
                  3980
82CE  FE06        3990 RADIUS:  CP   6              ; 6 Argumente?
82D0  C0          4000          RET  NZ             ; wenn nicht, zum BASIC
82D1  CD6085      4010          CALL DEG            ; Gradmaß setzen
82D4  CD7285      4020          CALL GRSTO          ; Grafikwerte speichern
82D7  CD1683      4030          CALL FARBE          ; Grafikstift setzen
82DA  CDD385      4040          CALL GTH10X         ; mittex holen
82DD  CDCC85      4050          CALL GTDE8X         ; mittey holen
82E0  EB          4060          EX   DE,HL          ; x und y tauschen
82E1  CD4685      4070          CALL MITTE          ; Mittelpunkt setzen
82E4  CDC585      4080          CALL GTHL6X         ; radiusx holen
82E7  11F285      4090          LD   DE,RADX        ; Adr. v. radiusx
82EA  CD3485      4100          CALL INTFLO         ; zu Float wandeln
82ED  CDBE85      4110          CALL GTDE4X         ; radiusy holen
82F0  21F785      4120          LD   HL,RADY        ; Adr. v. radiusy
82F3  EB          4130          EX   DE,HL          ; Adr. und Wert tauschen
82F4  CD3485      4140          CALL INTFLO         ; zu Float wandeln
82F7  CDB785      4150          CALL GTHL2X         ; winkel holen
82FA  11FC85      4160          LD   DE,WINKEL      ; Adr. v. winkel
82FD  CD3485      4170          CALL INTFLO         ; zu Float wandeln
8300  CDFD84      4180          CALL RECHNE         ; x und y berechnen
8303  2ABB80      4190          LD   HL,(Y_INT)     ; y holen
8306  ED5BC380    4200          LD   DE,(X_INT)     ; x holen
830A  CDF6BB      4210          CALL DRAW           ; DRAW x,y vom Ursprung
830D  CD8D85      4220          CALL GRREST         ; Grafik auf alte Werte
8310  C36C85      4230          JP   DEGRES         ; DEG/RAD herstellen
                  4240
00FF              4250          IF   CPC464         ; NUR für CPC 464
8313  FE01        4260 GRPEN:   CP   1              ; 1 Argument?
8315  C0          4270          RET  NZ             ; wenn nicht, zum BASIC
8316              4280          END                 ; NICHT für CPC 664/6128
8316  DD7E00      4290 FARBE:   LD   A,(IX+0)       ; Stiftnummer holen
8319  C3DEBB      4300          JP   SGRPEN         ; Grafikstift setzen
                  4310
00FF              4320          IF   CPC464         ; NUR für CPC464
831C  FE01        4330 GRPAP:   CP   1              ; 1 Argument?
831E  C0          4340          RET  NZ             ; wenn nicht, zum BASIC
831F  DD7E00      4350          LD   A,(IX+0)       ; Papernummer holen
8322  C3E4BB      4360          JP   SGRPAP         ; Grafikpaper setzen
8325              4370          END                 ; NICHT für CPC 664/6128
                  4380
8325  FE01        4390 GRMODE:  CP   1              ; 1 Argument?
8327  C0          4400          RET  NZ             ; wenn nicht, zum BASIC
8328  DD7E00      4410          LD   A,(IX+0)       ; Moduswert holen
832B  C359BC      4420          JP   SGRMOD         ; Grafikmodus setzen
                  4430
832E  FE07        4440 QUADER:  CP   7              ; 7 Argumente?
8330  C0          4450          RET  NZ             ; wenn nicht, zum BASIC
8331  CD7285      4460          CALL GRSTO          ; Grafikwerte speichern
8334  CD1683      4470          CALL FARBE          ; Grafikstift setzen
8337  CDDA85      4480          CALL GTD12X         ; x1 holen
```

```
833A  ED53C380  4490          LD   (X_INT),DE      ; für Rechteck speichern
833E  CDD385    4500          CALL GTH10X          ; y1 holen
8341  22BB80    4510          LD   (Y_INT),HL      ; für Rechteck speichern
8344  CDCC85    4520          CALL GTDE8X          ; x2 holen
8347  CDC585    4530          CALL GTHL6X          ; y2 holen
834A  D5        4540          PUSH DE              ; x2 zwischenspeichern
834B  E5        4550          PUSH HL              ; y2 zwischenspeichern
834C  CDB380    4560          CALL R_ECK           ; Rechteck x1,y1,x2,y2
834F  D1        4570          POP  DE              ; y2 holen
8350  2ABB80    4580          LD   HL,(Y_INT)      ; y1 holen
8353  CDA485    4590          CALL VERGL           ; vergleichen
8356  227584    4600          LD   (OBENY),HL      ; obeny speichern
8359  E5        4610          PUSH HL              ; und zwischenspeichern
835A  ED535784  4620          LD   (UNTENY),DE     ; unteny speichern
835E  CDB785    4630          CALL GTHL2X          ; untenhinteny holen
8361  CDAFBD    4640          CALL INTSB1          ; HL = untenhy - unteny
8364  22F084    4650          LD   (ALTY),HL       ; deltay speichern
8367  D1        4660          POP  DE              ; obeny holen
8368  CDACBD    4670          CALL INTADD          ; HL = obeny + deltay
836B  22BB80    4680          LD   (Y_INT),HL      ; für Rechteck speichern
836E  D1        4690          POP  DE              ; x2 holen
836F  2AC380    4700          LD   HL,(X_INT)      ; x1 holen
8372  CDA485    4710          CALL VERGL           ; vergleichen
8375  229884    4720          LD   (RECHTX),HL     ; rechtsx speichern
8378  E5        4730          PUSH HL              ; und zwischenspeichern
8379  ED538584  4740          LD   (LINKSX),DE     ; linksx speichern
837D  EB        4750          EX   DE,HL           ; linksx nach HL
837E  CDBE85    4760          CALL GTDE4X          ; linkshintenx holen
8381  CDB2BD    4770          CALL INTSB2          ; HL = linkshx - linksx
8384  22ED84    4780          LD   (ALTX),HL       ; deltax speichern
8387  D1        4790          POP  DE              ; rechtsx holen
8388  CDACBD    4800          CALL INTADD          ; HL = rechtsx + deltax
838B  22C380    4810          LD   (X_INT),HL      ; für Rechteck speichern
838E  CDBE85    4820          CALL GTDE4X          ; linkshintenx holen
8391  CDB785    4830          CALL GTHL2X          ; untenhinteny holen
8394  CDB380    4840          CALL R_ECK           ; hinteres Rechteck
8397  2A7584    4850          LD   HL,(OBENY)      ; obeny holen
839A  ED5B9884  4860          LD   DE,(RECHTX)     ; rechtsx holen
839E  E5        4870          PUSH HL              ; obeny speichern
839F  D5        4880          PUSH DE              ; rechtsx speichern
83A0  CDC0BB    4890          CALL MOVE            ; MOVE rechtsx,obeny
83A3  CDEC84    4900          CALL SCHRAEG         ; DRAWR deltax,deltay
83A6  D1        4910          POP  DE              ; rechtsx holen
83A7  2A5784    4920          LD   HL,(UNTENY)     ; unteny holen
83AA  E5        4930          PUSH HL              ; unteny speichern
83AB  CDC0BB    4940          CALL MOVE            ; MOVE rechtsx,unteny
83AE  CDEC84    4950          CALL SCHRAEG         ; DRAWR deltax,deltay
83B1  E1        4960          POP  HL              ; unteny holen
83B2  ED5B8584  4970          LD   DE,(LINKSX)     ; linksx holen
83B6  D5        4980          PUSH DE              ; linksx speichern
83B7  CDC0BB    4990          CALL MOVE            ; MOVE linksx,unteny
83BA  CDEC84    5000          CALL SCHRAEG         ; DRAWR deltax,deltay
83BD  D1        5010          POP  DE              ; linksx holen
83BE  E1        5020          POP  HL              ; obeny holen
83BF  CDC0BB    5030          CALL MOVE            ; MOVE linksx,obeny
83C2  CDEC84    5040          CALL SCHRAEG         ; DRAWR deltax,deltay
83C5  C38D85    5050          JP   GRREST          ; Grafik auf alte Werte
                5060
83C8  FE09      5070 VOLLQU:  CP   9               ; 9 Argumente?
83CA  C0        5080          RET  NZ              ; wenn nicht, zum BASIC
83CB  DD7E00    5090          LD   A,(IX+0)        ; Farbe3 holen
83CE  328084    5100          LD   (FARBE3),A      ; und setzen
83D1  DD7E02    5110          LD   A,(IX+2)        ; Farbe2 holen
83D4  326A84    5120          LD   (FARBE2),A      ; und setzen
83D7  DD7E04    5130          LD   A,(IX+4)        ; Farbe1 holen
83DA  326284    5140          LD   (FARBE1),A      ; und setzen
83DD  CD7285    5150          CALL GRSTO           ; Grafikwerte speichern
83E0  CD11BC    5160          CALL GTMODE          ; aktuellen MODE holen
83E3  3C        5170          INC  A               ; A = MODE + 1
83E4  47        5180          LD   B,A             ; in Zähler holen
83E5  3E08      5190          LD   A,8             ; Stepzahl*2 laden
83E7  0F        5200 ST_LP:   RRCA                 ; Stepzahl/2
83E8  10FD      5210          DJNZ ST_LP           ; sooft wie MODE+1
83EA  329384    5220          LD   (STEP),A        ; Stepzahl speichern
83ED  CDDA85    5230          CALL GTD12X          ; x1 holen
83F0  DD6611    5240          LD   H,(IX+17)       ; x2 in HL holen
83F3  DD6E10    5250          LD   L,(IX+16)
83F6  CDA485    5260          CALL VERGL           ; vergleichen
83F9  229884    5270          LD   (RECHTX),HL     ; rechtsx speichern
```

```
83FC  ED538584  5280          LD   (LINKSX),DE       ; linksx speichern
8400  E5        5290          PUSH HL                ; rechtsx retten
8401  EB        5300          EX   DE,HL             ; linksx nach HL
8402  CDCC85    5310          CALL GTDE8X            ; linkshintenx holen
8405  CDB2BD    5320          CALL INTSB2            ; deltax berechnen
8408  22ED84    5330          LD   (ALTX),HL         ; deltax speichern
840B  CDCABD    5340          CALL INTSGN            ; Vorzeichen deltax?
840E  E3        5350          EX   (SP),HL           ; deltax <--> rechtsx
840F  3809      5360          JR   C,NEG1            ; Sprung, wenn deltax<0
8411  ED53C380  5370          LD   (X_INT),DE        ; linksx speichern
8415  CDAFBD    5380          CALL INTSB1            ; rechtsx-linksx
8418  1806      5390          JR   POS1              ; dort weiter
841A  22C380    5400 NEG1:    LD   (X_INT),HL        ; rechtsx speichern
841D  CDB2BD    5410          CALL INTSB2            ; linksx-rechtsx
8420  22E484    5420 POS1:    LD   (DX2),HL          ; deltax2 speichern
8423  CDD385    5430          CALL GTH10X            ; y2 holen
8426  EB        5440          EX   DE,HL             ; y2 nach DE
8427  CDE185    5450          CALL GTH14X            ; y1 holen
842A  CDA485    5460          CALL VERGL             ; vergleichen
842D  227584    5470          LD   (OBENY),HL        ; obeny speichern
8430  ED535784  5480          LD   (UNTENY),DE       ; unteny speichern
8434  E5        5490          PUSH HL                ; obeny retten
8435  CDC585    5500          CALL GTHL6X            ; untenhinteny holen
8438  CDAFBD    5510          CALL INTSB1            ; deltay berechnen
843B  22F084    5520          LD   (ALTY),HL         ; deltay speichern
843E  CDCABD    5530          CALL INTSGN            ; Vorzeichen deltay?
8441  E1        5540          POP  HL                ; obeny holen
8442  3808      5550          JR   C,NEG2            ; wenn deltay<0
8444  22BB80    5560          LD   (Y_INT),HL        ; sonst y = obeny
8447  CDB2BD    5570          CALL INTSB2            ; unteny - obeny
844A  1807      5580          JR   POS2              ; dort weiter
844C  ED53BB80  5590 NEG2:    LD   (Y_INT),DE        ; y = unteny
8450  CDAFBD    5600          CALL INTSB1            ; obeny - unteny
8453  22D384    5610 POS2:    LD   (DY2),HL          ; deltay2 speichern
8456  210000    5620          LD   HL,#0000          ; unteny laden
8457            5630 UNTENY:  EQU  $-2               ; Argument v. LD HL,....
8459  E5        5640 QUAD_Y:  PUSH HL                ; unteny retten
845A  ED5BC380  5650          LD   DE,(X_INT)        ; entspr. x holen
845E  CDC0BB    5660          CALL MOVE              ; MOVE x,y
8461  3E00      5670          LD   A,#00             ; Farbe1 laden
8462            5680 FARBE1:  EQU  $-1               ; Argument v. LD A,..
8463  CDDEBB    5690          CALL SGRPEN            ; Grafikstift setzen
8466  CDE384    5700          CALL DRDX2             ; DRAWR deltax2,0
8469  3E00      5710          LD   A,#00             ; Farbe2 laden
846A            5720 FARBE2:  EQU  $-1               ; Argument v. LD A,..
846B  CDDEBB    5730          CALL SGRPEN            ; Grafikstift setzen
846E  CDEC84    5740          CALL SCHRAEG           ; DRAWR deltax,deltay
8471  E1        5750          POP  HL                ; laufendes y holen
8472  23        5760          INC  HL                ; Y-Wert + 2
8473  23        5770          INC  HL
8474  110000    5780          LD   DE,#0000          ; obeny laden
8475            5790 OBENY:   EQU  $-2               ; Argument v. LD DE,....
8477  EB        5800          EX   DE,HL             ; y <--> obeny
8478  AF        5810          XOR  A                 ; Carry löschen
8479  ED52      5820          SBC  HL,DE             ; y kleiner obeny?
847B  EB        5830          EX   DE,HL             ; obeny <--> y
847C  F25984    5840          JP   P,QUAD_Y          ; dann Sprung
847F  3E00      5850          LD   A,#00             ; Farbe3 laden
8480            5860 FARBE3:  EQU  $-1               ; Argument v. LD A,..
8481  CDDEBB    5870          CALL SGRPEN            ; Grafikstift setzen
8484  110000    5880          LD   DE,#0000          ; linksx laden
8485            5890 LINKSX:  EQU  $-2               ; Argument v. LD DE,....
8487  D5        5900 QUAD_X:  PUSH DE                ; laufendes x retten
8488  2ABB80    5910          LD   HL,(Y_INT)        ; entspr. y holen
848B  CDC0BB    5920          CALL MOVE              ; MOVE x,y
848E  CDEC84    5930          CALL SCHRAEG           ; DRAWR deltax,deltay
8491  D1        5940          POP  DE                ; x holen
8492  0600      5950          LD   B,#00             ; Stepzahl laden
8493            5960 STEP:    EQU  $-1               ; Argument v. LD B,..
8494  13        5970 STEPLP:  INC  DE                ; x = x + 1
8495  10FD      5980          DJNZ STEPLP            ; sooft wie Stepzahl
8497  210000    5990          LD   HL,#0000          ; rechtsx laden
8498            6000 RECHTX:  EQU  $-2               ; Argument v. LD HL,....
849A  AF        6010          XOR  A                 ; Carry löschen
849B  ED52      6020          SBC  HL,DE             ; x kleiner rechtsx?
849D  F28784    6030          JP   P,QUAD_X          ; ja, dann Sprung
84A0  E1        6040          POP  HL                ; deltax holen
84A1  CDCABD    6050          CALL INTSGN            ; Vorzeichen?
84A4  3805      6060          JR   C,NEG3            ; Sprung wenn <0
```

```
84A6  2A9884   6070          LD    HL,(RECHTX)      ; rechtsx in HL
84A9  1803     6080          JR    POS3             ; dort weiter
84AB  2A8584   6090 NEG3:    LD    HL,(LINKSX)      ; linksx in HL
84AE  E5       6100 POS3:    PUSH HL                ; x speichern
84AF  2AE484   6110          LD    HL,(DX2)         ; deltax2 laden
84B2  CDC7BD   6120          CALL  INTVZW           ; deltax2=-deltax2
84B5  22E484   6130          LD    (DX2),HL         ; abspeichern
84B8  AF       6140          XOR   A                ; A löschen
84B9  CDDEBB   6150          CALL  SGRPEN           ; Grafikstift 0
84BC  D1       6160          POP   DE               ; x holen
84BD  2ABB80   6170          LD    HL,(Y_INT)       ; y holen
84C0  D5       6180          PUSH DE                ; x retten
84C1  E5       6190          PUSH HL                ; y retten
84C2  CDC0BB   6200          CALL  MOVE             ; MOVE x,y
84C5  CDEC84   6210          CALL  SCHRAEG          ; DRAWR deltax,deltay
84C8  E1       6220          POP   HL               ; y holen
84C9  D1       6230          POP   DE               ; x holen
84CA  D5       6240          PUSH DE                ; x retten
84CB  E5       6250          PUSH HL                ; y retten
84CC  CDC0BB   6260          CALL  MOVE             ; MOVE x,y
84CF  110000   6270          LD    DE,#0000         ; x = 0
84D2  210000   6280          LD    HL,#0000         ; y laden
84D3           6290 DY2:     EQU   $-2              ; Argument v. LD HL,....
84D5  CDF9BB   6300          CALL  DRAWR            ; DRAWR 0,deltay2
84D8  E1       6310          POP   HL               ; y holen
84D9  D1       6320          POP   DE               ; x holen
84DA  CDC0BB   6330          CALL  MOVE             ; MOVE x,y
84DD  CDE384   6340          CALL  DRDX2            ; DRAWR deltax2,0
84E0  C38D85   6350          JP    GRREST           ; Grafik auf alte Werte
               6360
8552  C3C9BB   6950          JP    SETORG           ; mittelpunkt setzen
               6960
8555  3AF7B8   6970 RAD:     LD    A,(DEGREE)       ; DEG/RAD Flag laden
8558  326D85   6980          LD    (FLAG),A         ; und speichern
855B  AF       6990          XOR   A                ; Flag für RAD
855C  32F7B8   7000          LD    (DEGREE),A       ; laden
855F  C9       7010          RET                    ; und zurück
               7020
8560  3AF7B8   7030 DEG:     LD    A,(DEGREE)       ; DEG/RAD Flag holen
8563  326D85   7040          LD    (FLAG),A         ; und speichern
8566  3EFF     7050          LD    A,JA             ; Flag für DEG
8568  32F7B8   7060          LD    (DEGREE),A       ; laden
856B  C9       7070          RET                    ; und zurück
               7080
856C  3E00     7090 DEGRES:  LD    A,#00            ; alter DEGREE-Zustand
856D           7100 FLAG:    EQU   $-1              ; Argument v. LD A,..
856E  32F7B8   7110          LD    (DEGREE),A       ; wieder herstellen
8571  C9       7120          RET                    ; und Rückkehr
               7130
8572  CDCCBB   7140 GRSTO:   CALL  GETORG           ; Origin holen
8575  ED539385 7150          LD    (ORG_X),DE       ; X-Koord. speichern
8579  229685   7160          LD    (ORG_Y),HL       ; Y-Koord. speichern
857C  CDC6BB   7170          CALL  XY_POS           ; Grafikcursor holen
857F  ED539C85 7180          LD    (X_POS),DE       ; X-Koord. speichern
8583  229F85   7190          LD    (Y_POS),HL       ; Y-Koord. speichern
8586  CDE1BB   7200          CALL  GGRPEN           ; Grafikstift holen
8589  328E85   7210          LD    (GR_PEN),A       ; Grafikstift speichern
858C  C9       7220          RET                    ; zurück
               7230
858D  3E00     7240 GRREST:  LD    A,#00            ; Grafikstift laden
858E           7250 GR_PEN:  EQU   $-1              ; Argument v. LD A,..
858F  CDDEBB   7260          CALL  SGRPEN           ; Grafikstift setzen
8592  110000   7270          LD    DE,#0000         ; X-Koord. laden
8593           7280 ORG_X:   EQU   $-2              ; Argument v. LD DE,....
8595  210000   7290          LD    HL,#0000         ; Y-Koord. laden
8596           7300 ORG_Y:   EQU   $-2              ; Argument v. LD HL,....
8598  CDC9BB   7310          CALL  SETORG           ; Origin setzen
859B  110000   7320          LD    DE,#0000         ; X-Koord. laden
859C           7330 X_POS:   EQU   $-2              ; Argument v. LD DE,....
859E  210000   7340          LD    HL,#0000         ; Y-Koord. laden
859F           7350 Y_POS:   EQU   $-2              ; Argument v. LD HL,....
85A1  C3C0BB   7360          JP    MOVE             ; Grafikcursor setzen
               7370
85A4  CDC4BD   7380 VERGL:   CALL  INTVGL           ; HL vgl. DE
85A7  3001     7390          JR    NC,HLgrDE        ; wenn HL größer DE, ok
85A9  EB       7400          EX    DE,HL            ; sonst vertauschen
85AA  C9       7410 HLgrDE:  RET                    ; zurück
               7420
85AB  E5       7430 COPYHD:  PUSH HL                ; HL retten
```

```
85AC  D5        7440          PUSH DE          ; DE retten
85AD  C5        7450          PUSH BC          ; BC retten
85AE  010500    7460          LD   BC,5        ; Anzahl der Bytes
85B1  EDB0      7470          LDIR             ; (HL) --> (DE)
85B3  C1        7480          POP  BC          ; BC holen
85B4  D1        7490          POP  DE          ; DE holen
85B5  E1        7500          POP  HL          ; HL holen
85B6  C9        7510          RET              ; Rückkehr
                7520
84E3  110000    6370 DRDX2:   LD   DE,#0000    ; deltax2 laden
84E4            6380 DX2:     EQU  $-2         ; Argument v. LD DE,....
84E6  210000    6390          LD   HL,#0000    ; y = 0
84E9  C3F9BB    6400          JP   DRAWR       ; DRAWR deltax2,0
                6410
84EC  110000    6420 SCHRAE:  LD   DE,#0000    ; deltax laden
84ED            6430 ALTX:    EQU  $-2         ; Argument v. LD DE,....
84EF  210000    6440          LD   HL,#0000    ; deltay laden
84F0            6450 ALTY:    EQU  $-2         ; Argument v. LD HL,....
84F2  C3F9BB    6460          JP   DRAWR       ; DRAWR deltax,deltay
                6470
84F5  D5        6480 QUADRI:  PUSH DE          ; Adresse retten
84F6  CD3485    6490          CALL INTFLO      ; HL zu Float (DE)
84F9  D1        6500          POP  DE          ; Adresse holen => DE=HL
84FA  C361BD    6510          JP   FLOMUL      ; (HL) = Quadrat von HL
                6520
84FD  E5        6530 RECHNE:  PUSH HL          ; Adr. v. winkel retten
84FE  110186    6540          LD   DE,ZWIE     ; Adr. Zwischenspeicher
8501  CDAB85    6550          CALL COPYHD      ; winkel copieren
8504  EB        6560          EX   DE,HL       ; HL = Adr. v. zwie
8505  CD88BD    6570          CALL FLOSIN      ; zwie = sin(zwie)
8508  11F785    6580          LD   DE,RADY     ; Adr. v. radiusy
850B  CD61BD    6590          CALL FLOMUL      ; zwie=rady*sin(winkel)
850E  CD46BD    6600          CALL FLO_HL      ; zu Integer wandeln
8511  47        6610          LD   B,A         ; Vorzeichen in B holen
8512  FCA9BD    6620          CALL M,ISIGNB    ; Vorzeichen übernehmen
8515  22BB80    6630          LD   (Y_INT),HL  ; y speichern
8518  E1        6640          POP  HL          ; Adr. aktueller Winkel
8519  110186    6650          LD   DE,ZWIE     ; Adr. Zwischenspeicher
851C  CDAB85    6660          CALL COPYHD      ; winkel copieren
851F  EB        6670          EX   DE,HL       ; HL = Adr. v. zwie
8520  CD8BBD    6680          CALL FLOCOS      ; zwie = cos(winkel)
8523  11F285    6690          LD   DE,RADX     ; Adr. v. radiusx
8526  CD61BD    6700          CALL FLOMUL      ; zwie=radx*cos(winkel)
8529  CD46BD    6710          CALL FLO_HL      ; zu Integer wandeln
852C  47        6720          LD   B,A         ; Vorzeichen in B holen
852D  FCA9BD    6730          CALL M,ISIGNB    ; Vorzeichen übernehmen
8530  22C380    6740          LD   (X_INT),HL  ; x speichern
8533  C9        6750          RET              ; Rücksprung
                6760
8534  D5        6770 INTFLO:  PUSH DE          ; Zieladr. retten
8535  CDA3BD    6780          CALL INTABS      ; Betrag v. HL, B=Vorz.
8538  D1        6790          POP  DE          ; Adr. holen
8539  CD40BD    6800          CALL HL_FLO      ; zu Float wandeln
853C  78        6810          LD   A,B         ; Vorzeichen in A
853D  B7        6820          OR   A           ; und prüfen
853E  DDE5      6830          PUSH IX          ; IX für Param. retten
8540  FC6DBD    6840          CALL M,FLOVZW    ; HL<0 --> Vorz.wechsel
8543  DDE1      6850          POP  IX          ; IX für Param. holen
8545  C9        6860          RET              ; sonst Return
                6870
8546  E5        6880 MITTE:   PUSH HL          ; mittey retten
8547  2A9385    6890          LD   HL,(ORG_X)  ; x v. Origin laden
854A  19        6900          ADD  HL,DE       ; neues x f. Origin
854B  E3        6910          EX   (SP),HL     ; x retten, mittey holen
854C  ED5B9685  6920          LD   DE,(ORG_Y)  ; y v. Origin laden
8550  19        6930          ADD  HL,DE       ; neues y f. Origin
8551  D1        6940          POP  DE          ; x holen
85B7  DD6603    7530 GTHL2X:  LD   H,(IX+3)    ; LD HL,(IX+2)
85BA  DD6E02    7540          LD   L,(IX+2)
85BD  C9        7550          RET              ; Rückkehr
                7560
85BE  DD5605    7570 GTDE4X:  LD   D,(IX+5)    ; LD DE,(IX+4)
85C1  DD5E04    7580          LD   E,(IX+4)
85C4  C9        7590          RET              ; Rückkehr
                7600
85C5  DD6607    7610 GTHL6X:  LD   H,(IX+7)    ; LD HL,(IX+6)
85C8  DD6E06    7620          LD   L,(IX+6)
85CB  C9        7630          RET              ; Rückkehr
                7640
```

```
85CC  DD5609    7650 GTDE8X:  LD    D,(IX+9)           ; LD DE,(IX+8)
85CF  DD5E08    7660          LD    E,(IX+8)
85D2  C9        7670          RET                      ; Rückkehr
                7680
85D3  DD660B    7690 GTH10X:  LD    H,(IX+11)          ; LD HL,(IX+10)
85D6  DD6E0A    7700          LD    L,(IX+10)
85D9  C9        7710          RET                      ; Rückkehr
                7720
85DA  DD560D    7730 GTD12X:  LD    D,(IX+13)          ; LD DE,(IX+12)
85DD  DD5E0C    7740          LD    E,(IX+12)
85E0  C9        7750          RET                      ; Rückkehr
                7760
85E1  DD660F    7770 GTH14X:  LD    H,(IX+15)          ; LD HL,(IX+14)
85E4  DD6E0E    7780          LD    L,(IX+14)
85E7  C9        7790          RET                      ; Rückkehr
                7800
85E8  00000000  7810 EINS:    DEFB #00,#00,#00,#00,#81 ; Konstante 1
85ED            7820 Y^2:     DEFS 5
85F2            7830 RADX:    DEFS 5
85F7            7840 RADY:    DEFS 5
85FC            7850 WINKEL:  DEFS 5
8601            7860 ZWIE:    DEFS 5
8606            7870 WIEND:   DEFS 5
860B            7880 WISTEP:  DEFS 5

Pass 2 errors: 00

Table used:  1652  from  1700
```

```
60000 'BASICLOADER für die Grafikbefehle zum CPC 464
60010 IF HIMEM>32767 THEN MEMORY &7FFF
60020 IF PEEK(&8000)=&C9 THEN RETURN
60030 RESTORE 61000
60040 zeile=0
60050 FOR n%=&8000 TO &861F STEP 16
60060     zeile=zeile+1
60070     summe%=0
60080     FOR n1%=0 TO 15
60090         READ byte$
60100         byte%=VAL("&"+byte$)
60110         POKE n%+n1%,byte%
60120         summe%=summe%+byte%
60130     NEXT
60140     READ check$
60150     IF VAL("&"+check$)=summe% THEN 60180
60160        PRINT" Routinen nicht korrekt geladen !!!"
60170        PRINT"Fehler in der";zeile;". DATA-Zeile":END
60180 NEXT
60190 CALL &8000
60200 RETURN
61000 'DATA-Zeilen für die Grafikbefehle zum CPC 464
61001 DATA 01,0F,80,21,8D,80,CD,D1,BC,3E,C9,32,00,80,C9,38,06D2
61002 DATA 80,C3,91,80,C3,D4,80,C3,1D,81,C3,27,81,C3,16,82,0892
61003 DATA C3,3A,82,C3,42,82,C3,CE,82,C3,13,83,C3,1C,83,C3,0897
61004 DATA 25,83,C3,2E,83,C3,C8,83,52,45,43,48,54,45,43,CB,06F3
61005 DATA 42,4C,4F,43,CB,4B,52,45,49,D3,53,43,48,45,49,42,0597
61006 DATA C5,42,4F,47,45,CE,42,4F,47,45,4E,2E,C4,42,4F,47,05E5
61007 DATA 45,4E,2E,D2,52,41,44,49,55,D3,47,52,50,45,CE,47,061E
61008 DATA 52,50,41,50,45,D2,47,52,4D,4F,44,C5,51,55,41,44,05B3
61009 DATA 45,D2,56,4F,4C,4C,51,55,41,44,45,D2,00,00,00,00,0496
61010 DATA 00,FE,05,C0,CD,72,85,CD,16,83,CD,C5,85,22,BB,80,0861
61011 DATA CD,CC,85,ED,53,C3,80,CD,B7,85,CD,BE,85,CD,B3,80,0ABA
61012 DATA C3,8D,85,D5,E5,D5,CD,C0,BB,D1,21,00,00,E5,CD,F6,0A46
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung

```
61013 DATA BB,E1,11,00,00,D5,CD,F6,BB,D1,E1,E5,CD,F6,BB,E1,0AF6
61014 DATA D1,C3,F6,BB,FE,05,C0,CD,72,85,CD,16,83,CD,CC,85,0A50
61015 DATA ED,53,FF,80,CD,BE,85,ED,53,07,81,CD,B7,85,EB,CD,0A58
61016 DATA C5,85,CD,A4,85,22,BB,80,CB,83,2A,BB,80,D5,11,00,0836
61017 DATA 00,E5,CD,C0,BB,E1,11,00,00,E5,CD,F6,BB,E1,2B,2B,08B9
61018 DATA 22,BB,80,D1,B7,ED,52,F2,FA,80,C3,8D,85,FE,05,C0,0A28
61019 DATA 3E,FF,32,6D,85,18,07,FE,05,C0,AF,32,6D,85,CD,72,0755
61020 DATA 85,CD,16,83,CD,C5,85,CD,CC,85,CD,46,85,21,00,00,07D9
61021 DATA 22,ED,84,DD,E5,CD,BE,85,21,F2,85,EB,CD,F5,84,DD,0B0B
61022 DATA E1,CD,B7,85,E5,11,F7,85,CD,F5,84,E1,CD,A3,BD,CB,0B7B
61023 DATA 85,22,F0,84,22,BB,80,11,ED,85,CD,F5,84,11,F7,85,08CE
61024 DATA CD,64,BD,11,E8,85,CD,5E,BD,11,F2,85,CD,61,BD,CD,0994
61025 DATA 79,BD,CD,46,BD,22,C3,80,CD,97,81,2A,BB,80,7C,B5,08E6
61026 DATA 2B,2B,20,D0,C3,8D,85,2A,BB,80,ED,5B,C3,80,E5,D5,08C5
61027 DATA CD,EA,BB,3A,6D,85,B7,28,15,2A,F0,84,ED,5B,ED,84,08E9
61028 DATA E5,D5,CD,F6,BB,E1,CD,C7,BD,D1,EB,CD,EA,BB,E1,CD,0D46
61029 DATA C7,BD,D1,EB,CD,F6,BB,2A,BB,80,CD,C7,BD,ED,5B,C3,0B7F
61030 DATA 80,E5,D5,CD,EA,BB,3A,6D,85,B7,28,1A,2A,F0,84,CD,093C
61031 DATA C7,BD,ED,5B,ED,84,E5,D5,CD,F6,BB,E1,CD,C7,BD,D1,0C78
61032 DATA EB,CD,EA,BB,18,0A,2A,BB,80,7C,B5,20,03,E1,D1,C9,08B3
61033 DATA E1,CD,C7,BD,D1,EB,CD,F6,BB,2A,BB,80,22,F0,84,2A,0A91
61034 DATA C3,80,22,ED,84,C9,FE,08,C0,CD,B7,85,11,0B,86,CD,08DD
61035 DATA 34,85,CD,BE,85,21,06,86,EB,CD,34,85,CD,C5,85,11,080F
61036 DATA FC,85,CD,34,85,CD,60,85,18,2A,FE,08,C0,CD,60,85,0873
61037 DATA 18,06,FE,08,C0,CD,55,85,CD,B7,85,11,0B,86,CD,AB,07AE
61038 DATA 85,CD,BE,85,21,06,86,EB,CD,AB,85,CD,C5,85,11,FC,094E
61039 DATA 85,CD,AB,85,CD,72,85,CD,16,83,CD,E1,85,CD,DA,85,0A0B
61040 DATA EB,CD,46,85,CD,CC,85,21,F7,85,EB,CD,34,85,CD,D3,0A4F
61041 DATA 85,11,F2,85,CD,34,85,21,0B,86,CD,70,BD,28,39,38,06D8
61042 DATA 37,3E,FF,32,A5,82,21,FC,85,E5,CD,FD,84,ED,5B,C3,09AD
61043 DATA 80,2A,BB,80,3E,FF,B7,28,09,CD,C0,BB,AF,32,A5,82,085A
61044 DATA 18,03,CD,F6,BB,21,FC,85,11,0B,86,CD,58,BD,11,06,06D6
61045 DATA 86,EB,CD,6A,BD,E1,30,D1,CD,8D,85,C3,6C,85,FE,06,09DE
61046 DATA C0,CD,60,85,CD,72,85,CD,16,83,CD,D3,85,CD,CC,85,09DF
61047 DATA EB,CD,46,85,CD,C5,85,11,F2,85,CD,34,85,CD,BE,85,09B8
61048 DATA 21,F7,85,EB,CD,34,85,CD,B7,85,11,FC,85,CD,34,85,092F
61049 DATA CD,FD,84,2A,BB,80,ED,5B,C3,80,CD,F6,BB,CD,8D,85,0A9B
61050 DATA C3,6C,85,FE,01,C0,DD,7E,00,C3,DE,BB,FE,01,C0,DD,09C6
61051 DATA 7E,00,C3,E4,BB,FE,01,C0,DD,7E,00,C3,59,BC,FE,07,08D7
61052 DATA C0,CD,72,85,CD,16,83,CD,DA,85,ED,53,C3,80,CD,D3,0A39
61053 DATA 85,22,BB,80,CD,CC,85,CD,C5,85,D5,E5,CD,B3,80,D1,0AA2
61054 DATA 2A,BB,80,CD,A4,85,22,75,84,E5,ED,53,57,84,CD,B7,08FA
61055 DATA 85,CD,AF,BD,22,F0,84,D1,CD,AC,BD,22,BB,80,D1,2A,09B3
61056 DATA C3,80,CD,A4,85,22,98,84,E5,ED,53,85,84,EB,CD,BE,0A1B
61057 DATA 85,CD,B2,BD,22,ED,84,D1,CD,AC,BD,22,C3,80,CD,BE,0A4B
61058 DATA 85,CD,B7,85,CD,B3,80,2A,75,84,ED,5B,98,84,E5,D5,09CF
61059 DATA CD,C0,BB,CD,EC,84,D1,2A,57,84,E5,CD,C0,BB,CD,EC,0B41
61060 DATA 84,E1,ED,5B,85,84,D5,CD,C0,BB,CD,EC,84,D1,E1,CD,0B8F
61061 DATA C0,BB,CD,EC,84,C3,8D,85,FE,09,C0,DD,7E,00,32,80,0961
61062 DATA 84,DD,7E,02,32,6A,84,DD,7E,04,32,62,84,CD,72,85,073C
61063 DATA CD,11,BC,3C,47,3E,08,0F,10,FD,32,93,84,CD,DA,85,06F4
61064 DATA DD,66,11,DD,6E,10,CD,A4,85,22,98,84,ED,53,85,84,082C
61065 DATA E5,EB,CD,CC,85,CD,B2,BD,22,ED,84,CD,CA,BD,E3,38,0B2C
61066 DATA 09,ED,53,C3,80,CD,AF,BD,18,06,22,C3,80,CD,B2,BD,0884
61067 DATA 22,E4,84,CD,D3,85,EB,CD,E1,85,CD,A4,85,22,75,84,09DE
61068 DATA ED,53,57,84,E5,CD,C5,85,CD,AF,BD,22,F0,84,CD,CA,0A7D
61069 DATA BD,E1,38,08,22,BB,80,CD,B2,BD,18,07,ED,53,BB,80,0811
61070 DATA CD,AF,BD,22,D3,84,21,00,00,E5,ED,5B,C3,80,CD,C0,08D0
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

```
61071 DATA BB,3E,00,CD,DE,BB,CD,E3,84,3E,00,CD,DE,BB,CD,EC,09F0
61072 DATA 84,E1,23,23,11,00,00,EB,AF,ED,52,EB,F2,59,84,3E,078D
61073 DATA 00,CD,DE,BB,11,00,00,D5,2A,BB,80,CD,C0,BB,CD,EC,08B2
61074 DATA 84,D1,06,00,13,10,FD,21,00,00,AF,ED,52,F2,87,84,0687
61075 DATA E1,CD,CA,BD,38,05,2A,98,84,18,03,2A,85,84,E5,2A,0715
61076 DATA E4,84,CD,C7,BD,22,E4,84,AF,CD,DE,BB,D1,2A,BB,80,0A8E
61077 DATA D5,E5,CD,C0,BB,CD,EC,84,E1,D1,D5,E5,CD,C0,BB,11,0C04
61078 DATA 00,00,21,00,00,CD,F9,BB,E1,D1,CD,C0,BB,CD,E3,84,08D0
61079 DATA C3,8D,85,11,00,00,21,00,00,C3,F9,BB,11,00,00,21,04B0
61080 DATA 00,00,C3,F9,BB,D5,CD,34,85,D1,C3,61,BD,E5,11,01,087B
61081 DATA 86,CD,AB,85,EB,CD,88,BD,11,F7,85,CD,61,BD,CD,46,0A0B
61082 DATA BD,47,FC,A9,BD,22,BB,80,E1,11,01,86,CD,AB,85,EB,0924
61083 DATA CD,8B,BD,11,F2,85,CD,61,BD,CD,46,BD,47,FC,A9,BD,0A01
61084 DATA 22,C3,80,C9,D5,CD,A3,BD,D1,CD,40,BD,78,B7,DD,E5,0ABC
61085 DATA FC,6D,BD,DD,E1,C9,E5,2A,93,85,19,E3,ED,5B,96,85,0A33
61086 DATA 19,D1,C3,C9,BB,3A,F7,B8,32,6D,85,AF,32,F7,B8,C9,0997
61087 DATA 3A,F7,B8,32,6D,85,3E,FF,32,F7,B8,C9,3E,00,32,F7,085B
61088 DATA B8,C9,CD,CC,BB,ED,53,93,85,22,96,85,CD,C6,BB,ED,0AA5
61089 DATA 53,9C,85,22,9F,85,CD,E1,BB,32,8E,85,C9,3E,00,CD,083C
61090 DATA DE,BB,11,00,00,21,00,00,CD,C9,BB,11,00,00,21,00,044E
61091 DATA 00,C3,C0,BB,CD,C4,BD,30,01,EB,C9,E5,D5,C5,01,05,08F6
61092 DATA 00,ED,B0,C1,D1,E1,C9,DD,66,03,DD,6E,02,C9,DD,56,0968
61093 DATA 05,DD,5E,04,C9,DD,66,07,DD,6E,06,C9,DD,56,09,DD,078A
61094 DATA 5E,08,C9,DD,66,0B,DD,6E,0A,C9,DD,56,0D,DD,5E,0C,0722
61095 DATA C9,DD,66,0F,DD,6E,0E,C9,00,00,00,00,81,00,00,00,04BE
61096 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61097 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61098 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61099 'Ende der DATA-Zeilen für die Grafikbefehle zum CPC 464
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

```
60000 'BASICLOADER für die Grafikbefehle zum CPC 664
60010 IF HIMEM>32767 THEN MEMORY &7FFF
60020 IF PEEK(&8000)=&C9 THEN RETURN
60030 RESTORE 61000
60040 zeile=0
60050 FOR n%=&8000 TO &861F STEP 16
60060     zeile=zeile+1
60070     summe%=0
60080     FOR n1%=0 TO 15
60090         READ byte$
60100         byte%=VAL("&"+byte$)
60110         POKE n%+n1%,byte%
60120         summe%=summe%+byte%
60130     NEXT
60140     READ check$
60150     IF VAL("&"+check$)=summe% THEN 60180
60160        PRINT" Routinen nicht korrekt geladen !!!"
60170        PRINT"Fehler in der";zeile;". DATA-Zeile":END
60180 NEXT
60190 CALL &8000
60200 RETURN
61000 'DATA-Zeilen für die Grafikbefehle zum CPC 664
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

```
61001 DATA 01,0F,80,21,7B,80,CD,D1,BC,3E,C9,32,00,80,C9,32,06BA
61002 DATA 80,C3,9F,80,C3,E2,80,C3,2B,81,C3,35,81,C3,24,82,08D8
61003 DATA C3,48,82,C3,50,82,C3,DC,82,C3,27,83,C3,30,83,C3,08E9
61004 DATA CA,83,52,45,43,48,54,45,43,CB,42,4C,4F,43,CB,4B,064C
61005 DATA 52,45,49,D3,53,43,48,45,49,42,C5,42,4F,47,45,CE,0611
61006 DATA 42,4F,47,45,4E,2E,C4,42,4F,47,45,4E,2E,D2,52,41,055B
61007 DATA 44,49,55,D3,47,52,4D,4F,44,C5,51,55,41,44,45,D2,0635
61008 DATA 56,4F,4C,4C,51,55,41,44,45,D2,00,00,00,00,00,D7,0456
61009 DATA 2F,1D,C9,D7,3C,1D,C9,D7,4F,1D,C9,D7,58,1D,C9,D7,0806
61010 DATA 57,1D,C9,D7,07,DE,C9,D7,F2,DD,C9,D7,FE,DD,C9,FE,0BAA
61011 DATA 05,C0,CD,74,85,CD,21,83,CD,C7,85,22,C9,80,CD,CE,091B
61012 DATA 85,ED,53,D1,80,CD,B9,85,CD,C0,85,CD,C1,80,C3,8F,0A93
61013 DATA 85,D5,E5,D5,CD,C0,BB,D1,21,00,00,E5,CD,F6,BB,E1,0A92
61014 DATA 11,00,00,D5,CD,F6,BB,D1,E1,E5,CD,F6,BB,E1,D1,C3,0AEE
61015 DATA F6,BB,FE,05,C0,CD,74,85,CD,21,83,CD,CE,85,ED,53,0A0B
61016 DATA 0D,81,CD,C0,85,ED,53,15,81,CD,B9,85,EB,CD,C7,85,0985
61017 DATA CD,A6,85,22,C9,80,CB,83,2A,C9,80,D5,11,00,00,E5,07EF
61018 DATA CD,C0,BB,E1,11,00,00,E5,CD,F6,BB,E1,2B,2B,22,C9,08BF
61019 DATA 80,D1,B7,ED,52,F2,08,81,C3,8F,85,FE,05,C0,3E,FF,0999
61020 DATA 32,6F,85,18,07,FE,05,C0,AF,32,6F,85,CD,74,85,CD,0770
61021 DATA 21,83,CD,C7,85,CD,CE,85,CD,48,85,21,00,00,22,EF,07A9
61022 DATA 84,DD,E5,CD,C0,85,21,F4,85,EB,CD,F7,84,DD,E1,CD,0BB0
61023 DATA B9,85,E5,11,F9,85,CD,F7,84,E1,CD,7F,80,CB,85,22,0A19
61024 DATA F2,84,22,C9,80,11,EF,85,CD,F7,84,11,F9,85,CD,85,098F
61025 DATA BD,11,EA,85,CD,7F,BD,11,F4,85,CD,82,BD,CD,9A,BD,0A00
61026 DATA CD,67,BD,22,D1,80,CD,A5,81,2A,C9,80,7C,B5,2B,2B,0851
61027 DATA 20,D0,C3,8F,85,2A,C9,80,ED,5B,D1,80,E5,D5,CD,EA,0A44
61028 DATA BB,3A,6F,85,B7,28,15,2A,F2,84,ED,5B,EF,84,E5,D5,08F2
61029 DATA CD,F6,BB,E1,CD,97,80,D1,EB,CD,EA,BB,E1,CD,97,80,0C36
61030 DATA D1,EB,CD,F6,BB,2A,C9,80,CD,97,80,ED,5B,D1,80,E5,0B0F
61031 DATA D5,CD,EA,BB,3A,6F,85,B7,28,1A,2A,F2;84,CD,97,80,08F2
61032 DATA ED,5B,EF,84,E5,D5,CD,F6,BB,E1,CD,97,80,D1,EB,CD,0C41
61033 DATA EA,BB,18,0A,2A,C9,80,7C,B5,20,03,E1,D1,C9,E1,CD,08B7
61034 DATA 97,80,D1,EB,CD,F6,BB,2A,C9,80,22,F2,84,2A,D1,80,09D7
61035 DATA 22,EF,84,C9,FE,08,C0,CD,B9,85,11,0D,86,CD,36,85,085B
61036 DATA CD,C0,85,21,08,86,EB,CD,36,85,CD,C7,85,11,FE,85,08E1
61037 DATA CD,36,85,CD,62,85,18,2A,FE,08,C0,CD,62,85,18,06,0716
61038 DATA FE,08,C0,CD,57,85,CD,B9,85,11,0D,86,CD,AD,85,CD,08EA
61039 DATA C0,85,21,08,86,EB,CD,AD,85,CD,C7,85,11,FE,85,CD,0958
61040 DATA AD,85,CD,74,85,CD,21,83,CD,E3,85,CD,DC,85,EB,CD,0A84
61041 DATA 48,85,CD,CE,85,21,F9,85,EB,CD,36,85,CD,D5,85,11,0937
61042 DATA F4,85,CD,36,85,21,0D,86,CD,91,BD,28,39,38,37,3E,06DE
61043 DATA FF,32,B3,82,21,FE,85,E5,CD,FF,84,ED,5B,D1,80,2A,0A02
61044 DATA C9,80,3E,FF,B7,28,09,CD,C0,BB,AF,32,B3,82,18,03,07E7
61045 DATA CD,F6,BB,21,FE,85,11,0D,86,CD,79,BD,11,08,86,EB,0853
61046 DATA CD,8B,BD,E1,30,D1,CD,8F,85,C3,6E,85,FE,06,C0,CD,0A1F
61047 DATA 62,85,CD,74,85,CD,21,83,CD,D5,85,CD,CE,85,EB,CD,0A1D
61048 DATA 48,85,CD,C7,85,11,F4,85,CD,36,85,CD,C0,85,21,F9,0924
61049 DATA 85,EB,CD,36,85,CD,B9,85,11,FE,85,CD,36,85,CD,FF,09EB
61050 DATA 84,2A,C9,80,ED,5B,D1,80,CD,F6,BB,CD,8F,85,C3,6E,0A20
61051 DATA 85,DD,7E,00,C3,DE,BB,FE,01,C0,DD,7E,00,C3,59,BC,092E
61052 DATA FE,07,C0,CD,74,85,CD,21,83,CD,DC,85,ED,53,D1,80,09BB
61053 DATA CD,D5,85,22,C9,80,CD,CE,85,CD,C7,85,D5,E5,CD,C1,0B13
61054 DATA 80,D1,2A,C9,80,CD,A6,85,22,77,84,E5,ED,53,59,84,08DB
61055 DATA CD,B9,85,CD,8B,80,22,F2,84,D1,CD,87,80,22,C9,80,098B
61056 DATA D1,2A,D1,80,CD,A6,85,22,9A,84,E5,ED,53,87,84,EB,099F
61057 DATA CD,C0,85,CD,8F,80,22,EF,84,D1,CD,87,80,22,D1,80,099B
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

```
61058 DATA CD,C0,85,CD,B9,85,CD,C1,80,2A,77,84,ED,5B,9A,84,09B6
61059 DATA E5,D5,CD,C0,BB,CD,EE,84,D1,2A,59,84,E5,CD,C0,BB,0B46
61060 DATA CD,EE,84,E1,ED,5B,87,84,D5,CD,C0,BB,CD,EE,84,D1,0BA0
61061 DATA E1,CD,C0,BB,CD,EE,84,C3,8F,85,FE,09,C0,DD,7E,00,0A61
61062 DATA 32,82,84,DD,7E,02,32,6C,84,DD,7E,04,32,64,84,CD,06FD
61063 DATA 74,85,CD,11,BC,3C,47,3E,08,0F,10,FD,32,95,84,CD,0690
61064 DATA DC,85,DD,66,11,DD,6E,10,CD,A6,85,22,9A,84,ED,53,0888
61065 DATA 87,84,E5,EB,CD,CE,85,CD,8F,80,22,EF,84,CD,9B,80,0A54
61066 DATA E3,38,09,ED,53,D1,80,CD,8B,80,18,06,22,D1,80,CD,07EB
61067 DATA 8F,80,22,E6,84,CD,D5,85,EB,CD,E3,85,CD,A6,85,22,09FC
61068 DATA 77,84,ED,53,59,84,E5,CD,C7,85,CD,8B,80,22,F2,84,0986
61069 DATA CD,9B,80,E1,38,08,22,C9,80,CD,8F,80,18,07,ED,53,07AF
61070 DATA C9,80,CD,8B,80,22,D5,84,21,00,00,E5,ED,5B,D1,80,083B
61071 DATA CD,C0,BB,3E,00,CD,DE,BB,CD,E5,84,3E,00,CD,DE,BB,09C6
61072 DATA CD,EE,84,E1,23,23,11,00,00,EB,AF,ED,52,EB,F2,5B,0888
61073 DATA 84,3E,00,CD,DE,BB,11,00,00,D5,2A,C9,80,CD,C0,BB,07C9
61074 DATA CD,EE,84,D1,06,00,13,10,FD,21,00,00,AF,ED,52,F2,0737
61075 DATA 89,84,E1,CD,9B,80,38,05,2A,9A,84,18,03,2A,87,84,06AB
61076 DATA E5,2A,E6,84,CD,97,80,22,E6,84,AF,CD,DE,BB,D1,2A,09F9
61077 DATA C9,80,D5,E5,CD,C0,BB,CD,EE,84,E1,D1,D5,E5,CD,C0,0C83
61078 DATA BB,11,00,00,21,00,00,CD,F9,BB,E1,D1,CD,C0,BB,CD,0835
61079 DATA E5,84,C3,8F,85,11,00,00,21,00,00,C3,F9,BB,11,00,05FA
61080 DATA 00,21,00,00,C3,F9,BB,D5,CD,36,85,D1,C3,82,BD,E5,08AD
61081 DATA 11,03,86,CD,AD,85,EB,CD,A9,BD,11,F9,85,CD,82,BD,0952
61082 DATA CD,67,BD,47,FC,83,80,22,C9,80,E1,11,03,86,CD,AD,0897
61083 DATA 85,EB,CD,AC,BD,11,F4,85,CD,82,BD,CD,67,BD,47,FC,0A70
61084 DATA 83,80,22,D1,80,C9,D5,CD,7F,80,D1,CD,61,BD,78,B7,09CB
61085 DATA DD,E5,FC,8E,BD,DD,E1,C9,E5,2A,95,85,19,E3,ED,5B,0AFD
61086 DATA 98,85,19,D1,C3,C9,BB,3A,13,B1,32,6F,85,AF,32,13,0766
61087 DATA B1,C9,3A,13,B1,32,6F,85,3E,FF,32,13,B1,C9,3E,00,06D8
61088 DATA 32,13,B1,C9,CD,CC,BB,ED,53,95,85,22,98,85,CD,C6,093F
61089 DATA BB,ED,53,9E,85,22,A1,85,CD,E1,BB,32,90,85,C9,3E,091D
61090 DATA 00,CD,DE,BB,11,00,00,21,00,00,CD,C9,BB,11,00,00,04FA
61091 DATA 21,00,00,C3,C0,BB,CD,93,80,30,01,EB,C9,E5,D5,C5,08A3
61092 DATA 01,05,00,ED,B0,C1,D1,E1,C9,DD,66,03,DD,6E,02,C9,083B
61093 DATA DD,56,05,DD,5E,04,C9,DD,66,07,DD,6E,06,C9,DD,56,07D7
61094 DATA 09,DD,5E,08,C9,DD,66,0B,DD,6E,0A,C9,DD,56,0D,DD,079E
61095 DATA 5E,0C,C9,DD,66,0F,DD,6E,0E,C9,00,00,00,00,81,00,0528
61096 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61097 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61098 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61099 'Ende der DATA-Zeilen für die Grafikbefehle zum CPC 664
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

```
60000 'BASICLOADER für die Grafikbefehle zum CPC 6128
60010 IF HIMEM>32767 THEN MEMORY &7FFF
60020 IF PEEK(&8000)=&C9 THEN RETURN
60030 RESTORE 61000
60040 zeile=0
60050 FOR n%=&8000 TO &861F STEP 16
60060     zeile=zeile+1
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

```
60070       summe%=0
60080       FOR n1%=0 TO 15
60090           READ byte$
60100           byte%=VAL("&"+byte$)
60110           POKE n%+n1%,byte%
60120           summe%=summe%+byte%
60130       NEXT
60140       READ check$
60150       IF VAL("&"+check$)=summe% THEN 60180
60160          PRINT" Routinen nicht korrekt geladen !!!"
60170          PRINT"Fehler in der";zeile;". DATA-Zeile":END
60180 NEXT
60190 CALL &8000
60200 RETURN
61000 'DATA-Zeilen für die Grafikbefehle zum CPC 6128
61001 DATA 01,0F,80,21,7B,80,CD,D1,BC,3E,C9,32,00,80,C9,32,06BA
61002 DATA 80,C3,9F,80,C3,E2,80,C3,2B,81,C3,35,81,C3,24,82,08D8
61003 DATA C3,48,82,C3,50,82,C3,DC,82,C3,27,83,C3,30,83,C3,08E9
61004 DATA CA,83,52,45,43,48,54,45,43,CB,42,4C,4F,43,CB,4B,064C
61005 DATA 52,45,49,D3,53,43,48,45,49,42,C5,42,4F,47,45,CE,0611
61006 DATA 42,4F,47,45,4E,2E,C4,42,4F,47,45,4E,2E,D2,52,41,055B
61007 DATA 44,49,55,D3,47,52,4D,4F,44,C5,51,55,41,44,45,D2,0635
61008 DATA 56,4F,4C,4C,51,55,41,44,45,D2,00,00,00,00,00,D7,0456
61009 DATA 2A,1D,C9,D7,37,1D,C9,D7,4A,1D,C9,D7,53,1D,C9,D7,07F2
61010 DATA 52,1D,C9,D7,02,1E,C9,D7,ED,1D,C9,D7,F9,1D,C9,FE,0956
61011 DATA 05,C0,CD,74,85,CD,21,83,CD,C7,85,22,C9,80,CD,CE,091B
61012 DATA 85,ED,53,D1,80,CD,B9,85,CD,C0,85,CD,C1,80,C3,8F,0A93
61013 DATA 85,D5,E5,D5,CD,C0,BB,D1,21,00,00,E5,CD,F6,BB,E1,0A92
61014 DATA 11,00,00,D5,CD,F6,BB,D1,E1,E5,CD,F6,BB,E1,D1,C3,0AEE
61015 DATA F6,BB,FE,05,C0,CD,74,85,CD,21,83,CD,CE,85,ED,53,0A0B
61016 DATA 0D,81,CD,C0,85,ED,53,15,81,CD,B9,85,EB,CD,C7,85,0985
61017 DATA CD,A6,85,22,C9,80,CB,83,2A,C9,80,D5,11,00,00,E5,07EF
61018 DATA CD,C0,BB,E1,11,00,00,E5,CD,F6,BB,E1,2B,2B,22,C9,08BF
61019 DATA 80,D1,B7,ED,52,F2,08,81,C3,8F,85,FE,05,C0,3E,FF,0999
61020 DATA 32,6F,85,18,07,FE,05,C0,AF,32,6F,85,CD,74,85,CD,0770
61021 DATA 21,83,CD,C7,85,CD,CE,85,CD,48,85,21,00,00,22,EF,07A9
61022 DATA 84,DD,E5,CD,C0,85,21,F4,85,EB,CD,F7,84,DD,E1,CD,0BB0
61023 DATA B9,85,E5,11,F9,85,CD,F7,84,E1,CD,7F,80,CB,85,22,0A19
61024 DATA F2,84,22,C9,80,11,EF,85,CD,F7,84,11,F9,85,CD,88,0992
61025 DATA BD,11,EA,85,CD,82,BD,11,F4,85,CD,85,BD,CD,9D,BD,0A09
61026 DATA CD,6A,BD,22,D1,80,CD,A5,81,2A,C9,80,7C,B5,2B,2B,0854
61027 DATA 20,D0,C3,8F,85,2A,C9,80,ED,5B,D1,80,E5,D5,CD,EA,0A44
61028 DATA BB,3A,6F,85,B7,28,15,2A,F2,84,ED,5B,EF,84,E5,D5,08F2
61029 DATA CD,F6,BB,E1,CD,97,80,D1,EB,CD,EA,BB,E1,CD,97,80,0C36
61030 DATA D1,EB,CD,F6,BB,2A,C9,80,CD,97,80,ED,5B,D1,80,E5,0B0F
61031 DATA D5,CD,EA,BB,3A,6F,85,B7,28,1A,2A,F2,84,CD,97,80,08F2
61032 DATA ED,5B,EF,84,E5,D5,CD,F6,BB,E1,CD,97,80,D1,EB,CD,0C41
61033 DATA EA,BB,18,0A,2A,C9,80,7C,B5,20,03,E1,D1,C9,E1,CD,08B7
61034 DATA 97,80,D1,EB,CD,F6,BB,2A,C9,80,22,F2,84,2A,D1,80,09D7
61035 DATA 22,EF,84,C9,FE,08,C0,CD,B9,85,11,0D,86,CD,36,85,085B
61036 DATA CD,C0,85,21,08,86,EB,CD,36,85,CD,C7,85,11,FE,85,08E1
61037 DATA CD,36,85,CD,62,85,18,2A,FE,08,C0,CD,62,85,18,06,0716
61038 DATA FE,08,C0,CD,57,85,CD,B9,85,11,0D,86,CD,AD,85,CD,08EA
61039 DATA C0,85,21,08,86,EB,CD,AD,85,CD,C7,85,11,FE,85,CD,0958
61040 DATA AD,85,CD,74,85,CD,21,83,CD,E3,85,CD,DC,85,EB,CD,0A84
61041 DATA 48,85,CD,CE,85,21,F9,85,EB,CD,36,85,CD,D5,85,11,0937
61042 DATA F4,85,CD,36,85,21,0D,86,CD,94,BD,28,39,38,37,3E,06E1
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

```
61043 DATA FF,32,B3,82,21,FE,85,E5,CD,FF,84,ED,5B,D1,80,2A,0A02
61044 DATA C9,80,3E,FF,B7,28,09,CD,C0,BB,AF,32,B3,82,18,03,07E7
61045 DATA CD,F6,BB,21,FE,85,11,0D,86,CD,7C,BD,11,08,86,EB,0856
61046 DATA CD,8E,BD,E1,30,D1,CD,8F,85,C3,6E,85,FE,06,C0,CD,0A22
61047 DATA 62,85,CD,74,85,CD,21,83,CD,D5,85,CD,CE,85,EB,CD,0A1D
61048 DATA 48,85,CD,C7,85,11,F4,85,CD,36,85,CD,C0,85,21,F9,0924
61049 DATA 85,EB,CD,36,85,CD,B9,85,11,FE,85,CD,36,85,CD,FF,09EB
61050 DATA 84,2A,C9,80,ED,5B,D1,80,CD,F6,BB,CD,8F,85,C3,6E,0A20
61051 DATA 85,DD,7E,00,C3,DE,BB,FE,01,C0,DD,7E,00,C3,59,BC,092E
61052 DATA FE,07,C0,CD,74,85,CD,21,83,CD,DC,85,ED,53,D1,80,09BB
61053 DATA CD,D5,85,22,C9,80,CD,CE,85,CD,C7,85,D5,E5,CD,C1,0B13
61054 DATA 80,D1,2A,C9,80,CD,A6,85,22,77,84,E5,ED,53,59,84,08DB
61055 DATA CD,B9,85,CD,8B,80,22,F2,84,D1,CD,87,80,22,C9,80,098B
61056 DATA D1,2A,D1,80,CD,A6,85,22,9A,84,E5,ED,53,87,84,EB,099F
61057 DATA CD,C0,85,CD,8F,80,22,EF,84,D1,CD,87,80,22,D1,80,099B
61058 DATA CD,C0,85,CD,B9,85,CD,C1,80,2A,77,84,ED,5B,9A,84,09B6
61059 DATA E5,D5,CD,C0,BB,CD,EE,84,D1,2A,59,84,E5,CD,C0,BB,0B46
61060 DATA CD,EE,84,E1,ED,5B,87,84,D5,CD,C0,BB,CD,EE,84,D1,0BA0
61061 DATA E1,CD,C0,BB,CD,EE,84,C3,8F,85,FE,09,C0,DD,7E,00,0A61
61062 DATA 32,82,84,DD,7E,02,32,6C,84,DD,7E,04,32,64,84,CD,06FD
61063 DATA 74,85,CD,11,BC,3C,47,3E,08,0F,10,FD,32,95,84,CD,0690
61064 DATA DC,85,DD,66,11,DD,6E,10,CD,A6,85,22,9A,84,ED,53,0888
61065 DATA 87,84,E5,EB,CD,CE,85,CD,8F,80,22,EF,84,CD,9B,80,0A54
61066 DATA E3,38,09,ED,53,D1,80,CD,8B,80,18,06,22,D1,80,CD,07EB
61067 DATA 8F,80,22,E6,84,CD,D5,85,EB,CD,E3,85,CD,A6,85,22,09FC
61068 DATA 77,84,ED,53,59,84,E5,CD,C7,85,CD,8B,80,22,F2,84,0986
61069 DATA CD,9B,80,E1,38,08,22,C9,80,CD,8F,80,18,07,ED,53,07AF
61070 DATA C9,80,CD,8B,80,22,D5,84,21,00,00,E5,ED,5B,D1,80,083B
61071 DATA CD,C0,BB,3E,00,CD,DE,BB,CD,E5,84,3E,00,CD,DE,BB,09C6
61072 DATA CD,EE,84,E1,23,23,11,00,00,EB,AF,ED,52,EB,F2,5B,0888
61073 DATA 84,3E,00,CD,DE,BB,11,00,00,D5,2A,C9,80,CD,C0,BB,07C9
61074 DATA CD,EE,84,D1,06,00,13,10,FD,21,00,00,AF,ED,52,F2,0737
61075 DATA 89,84,E1,CD,9B,80,38,05,2A,9A,84,18,03,2A,87,84,06AB
61076 DATA E5,2A,E6,84,CD,97,80,22,E6,84,AF,CD,DE,BB,D1,2A,09F9
61077 DATA C9,80,D5,E5,CD,C0,BB,CD,EE,84,E1,D1,D5,E5,CD,C0,0C83
61078 DATA BB,11,00,00,21,00,00,CD,F9,BB,E1,D1,CD,C0,BB,CD,0835
61079 DATA E5,84,C3,8F,85,11,00,00,21,00,00,C3,F9,BB,11,00,05FA
61080 DATA 00,21,00,00,C3,F9,BB,D5,CD,36,85,D1,C3,85,BD,E5,08B0
61081 DATA 11,03,86,CD,AD,85,EB,CD,AC,BD,11,F9,85,CD,85,BD,0958
61082 DATA CD,6A,BD,47,FC,83,80,22,C9,80,E1,11,03,86,CD,AD,089A
61083 DATA 85,EB,CD,AF,BD,11,F4,85,CD,85,BD,CD,6A,BD,47,FC,0A79
61084 DATA 83,80,22,D1,80,C9,D5,CD,7F,80,D1,CD,64,BD,78,B7,09CE
61085 DATA DD,E5,FC,91,BD,DD,E1,C9,E5,2A,95,85,19,E3,ED,5B,0B00
61086 DATA 98,85,19,D1,C3,C9,BB,3A,13,B1,32,6F,85,AF,32,13,0766
61087 DATA B1,C9,3A,13,B1,32,6F,85,3E,FF,32,13,B1,C9,3E,00,06D8
61088 DATA 32,13,B1,C9,CD,CC,BB,ED,53,95,85,22,98,85,CD,C6,093F
61089 DATA BB,ED,53,9E,85,22,A1,85,CD,E1,BB,32,90,85,C9,3E,091D
61090 DATA 00,CD,DE,BB,11,00,00,21,00,00,CD,C9,BB,11,00,00,04FA
61091 DATA 21,00,00,C3,C0,BB,CD,93,80,30,01,EB,C9,E5,D5,C5,08A3
61092 DATA 01,05,00,ED,B0,C1,D1,E1,C9,DD,66,03,DD,6E,02,C9,083B
61093 DATA DD,56,05,DD,5E,04,C9,DD,66,07,DD,6E,06,C9,DD,56,07D7
61094 DATA 09,DD,5E,08,C9,DD,66,0B,DD,6E,0A,C9,DD,56,0D,DD,079E
61095 DATA 5E,0C,C9,DD,66,0F,DD,6E,0E,C9,00,00,00,00,81,00,0528
61096 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61097 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61098 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,0000
61099 'Ende der DATA-Zeilen für die Grafikbefehle zum CPC 6128
```

Programm 2.29: Der BASIC-Lader für die Grafikerweiterung (Forts.)

Programmbeschreibung

Für die Leser, die sich nicht für die Funktionsweise der Routinen interessieren und sie nur anwenden möchten, stehen zwei Möglichkeiten der Befehlseinbindung zur Verfügung. Die erste wäre das Eintippen des für ihre CPC Version passenden, abgedruckten BASIC-Laders und sein Aufruf mit

```
GOSUB 60000
```

wodurch die Lade-Routine des Laders die 1,5 KByte Maschinenprogramm in den Speicher schreibt und aufruft. Der Lader initialisiert die Routinen mit einem Sprung nach &8000.

Der zweite Weg geht über das Abtippen des (für den CPC 464 assemblierten) Assemblerlistings in einen geeigneten Assembler. Dies hat den Vorteil, daß man durch Änderung der ORG-Anweisung die Routinen in einem anderen Speicherbereich unterbringen kann. Damit korrekt assembliert werden kann, muß eines der Labels CPC 464, CPC 664 oder CPC 128 die Zuweisung von JA erhalten. Die beiden anderen müssen auf NEIN stehen. Nach der Assemblierung muß das Maschinenprogramm durch einen Sprung an die in der ORG-Anweisung stehenden Adresse initialisiert werden. Im Normalfall geschieht dies mit

```
CALL &8000
```

und kann mehrmals erfolgen, da die Initialisierung den Einsprung mit RET belegt.

Damit die Befehle nach einem Aus- und wieder Einschalten nicht jedesmal mit dem Lader oder dem Assembler neu erstellt werden müssen, kann man das Maschinenprogramm mit dem Befehl

```
SAVE “GRABEF“,B,&8000,&620
```

speichern. Die Abspeicherung muß vor der Initialisierung durch CALL &8000 erfolgen oder zumindest das erste Byte mit POKE &8000,&01 korrigiert werden. So abgespeichert können die Befehle jederzeit in den Speicher geladen und initialisiert werden. Dazu ist dann die Befehlsfolge

```
MEMORY &7FFF : LOAD “GRABEF“ : CALL &8000
```

– sofern der Name GRABEF gewählt wurde – notwendig.

Die Routinen wurden mit Absicht ab der Adresse &8000 untergebracht. Nur dadurch ist es möglich, z. B. das Spriteprogramm, die Grafikbefehle und die

Hardcopy zur gleichen Zeit im Speicher zu haben. Außerdem stehen noch etwa 2,5 KByte nach den Grafikbefehlen zur Ergänzung mit weiteren Befehlen oder zur Verbesserung zur Verfügung. Wer aber möglichst viel Speicher benötigt, kann, wie weiter oben schon angegeben, durch eine andere ORG-Anweisung die Grafikbefehle z. B. ab der Adresse &A000 assemblieren lassen.

Weiter geht's für diejenigen, die sich für die Funktionsweise der Grafik-Routinen interessieren.

Allgemeine Erläuterungen zum Listing

In den Zeilen 10 bis 200 und 650 bis 1350 werden sogenannte LABELs definiert. Labels sind nichts anderes als Konstanten, die Integer-Werte annehmen können. Sie werden dazu benutzt, um ein Assemblerprogramm übersichtlicher zu gestalten. Ein zweiter Vorteil ist, daß das Programm unabhängig von seiner Lage im Speicher ist. Erfolgt z. B. ein Sprung zu dem Label R_ECK, so errechnet der Assembler die zugehörige Adresse und setzt sie überall, wo das Label R_ECK als Operand vorkommt, ein.

Das Label in Zeile 80 hat die Bezeichnung ERWEIT und repräsentiert die Adresse der Betriebssystem-Routine, die eine Befehlserweiterung in das System einbindet. Dieser Routine muß im HL-Register die Adresse eines vier Byte langen ungenutzten Speicherbereichs zwischen &4000 und &BFFF übergeben werden. Im BC-Register steht dabei die Adresse der Sprungtabelle.

Die Funktion der Labels in den Zeilen 90 bis 180 geht – wenn nicht aus dem Namen – aus der Beschreibung der Grafik-Routinen im Kapitel 1 hervor.

Ein Sprung zu GTMODE (ROM-Routine), also &BC11, liefert im Akku den aktuellen Bildschirmmodus zurück. Der Akkuinhalt entspricht dem Modus (0,1,2). Bei MODE 0 ist zudem das Zero-Flag zurückgesetzt und das Carry-Flag gesetzt. Bei MODE 1 ist das Carry-Flag zurückgesetzt und das Zero-Flag gesetzt, und schließlich sind bei MODE 2 die beiden Flags zurückgesetzt.

Die Routine &BC59, also SGRMOD, setzt den Grafikmodus auf den im Akku übergebenen Modus. 0 entspricht dem überschreibenden, 1 dem EXCLUSIV-ODER verknüpfenden, 2 dem UND verknüpfenden und 3 dem ODER verknüpfenden Grafikschreibmodus. Der Akkuinhalt wird mit &03 UND-verknüpft, um nur die Werte zwischen 0 und 3 auszuwerten.

Mathematische Routinen

Die Routinen in den Zeilen 650 bis 1350 stellen Fließkomma-Routinen dar, die sonst nur von BASIC genutzt werden. Je nachdem, welchem der Labels CPC 464, CPC 664 und CPC 128 der Wert JA und den beiden anderen der

Wert NEIN in den Zeilen 40 bis 60 zugewiesen wurde, wird die für diesen CPC zutreffende Label- bzw. Sprungtabelle assembliert und die beiden anderen ausgelassen. Bei den Fließkommaoperationen ist nötig, daß alle Routinen, die Fließkommazahlen benutzen, diese mit Adressen im Bereich von &4000 und &BFFF ablegen. Dazu müssen übergebene Fließkommavariablen – wenn nötig – erst einmal in höhere Speicherbereiche kopiert werden. Die meisten Fließkomma-Routinen benutzen die Indexregister. Wenn also erst etwas berechnet wird und dann mit dem IX-Register ein Parameter geholt werden soll, so muß das IX-Register zwischengespeichert werden.

Die Routinen, die mit einem I bzw. INT beginnen, stellen die Integer-Routinen des BASIC dar und arbeiten hauptsächlich mit den Registern HL und DE, manchmal auch mit A und B. Da die Integer-Routinen bei den CPC 664 und CPC 6128 im BASIC-ROM liegen und von der Sprungtabelle verbannt wurden, mußten für diese Computer die Routinen mittels einer zusätzlichen Sprungtabelle zugänglich gemacht werden. Diese Sprungtabelle wird mit den DEFB-Anweisungen erzeugt.

Alle mathematischen Routinen, bis auf die Signums- und Vergleichsfunktionen, kehren mit gesetztem Carry-Flag zurück, wenn die Operation korrekt ausgeführt wurde.

HL FLO wandelt die im HL-Register übergebene Integer-Zahl in eine Fließkommazahl ab der im DE-Register übergebenen Adresse um.

Die Routine FLO HL wandelt die unter der Adresse im HL-Register gefundene Fließkommazahl in eine Integer-Zahl um, die im HL-Register zurückgeliefert wird.

Die mit FLOADD bezeichnete Routine addiert zwei Fließkommawerte. Der durch DE adressierte Wert wird zum durch HL adressierten addiert. Das Ergebnis ist unter der Adresse HL zu finden.

Die Routine FLOSB2 subtrahiert den durch HL adressierten Wert von dem durch DE adressierten. Das Ergebnis wird wieder unter der Adresse HL abgelegt.

FLOMUL multipliziert die durch DE und HL adressierten Werte und legt das Ergebnis unter der Adresse HL ab.

FLODIV hingegen dividiert den durch HL adressierten Wert durch den mit DE adressierten. Das Ergebnis der Routine wird wieder unter der Adresse HL abgelegt.

Die Routine FLOVGL vergleicht die durch HL und DE adressierten Werte. Ist der durch HL adressierte größer, so werden alle Flags zurückgesetzt, und

der Akku enthält den Wert 1. Sind beide Werte gleich, so ist der Akku 0 und das Zero-Flag gesetzt. Ist der durch HL adressierte Wert kleiner, so steht im Akku &FF, und es sind das Carry- und das Vorzeichen-Flag gesetzt.

FLOVZW, die Routine wechselt bei dem durch HL adressierten Wert das Vorzeichen.

Die Quadratwurzel des durch HL adressierten Wertes erreicht man durch Aufruf der Routine FLOSQR. Das Ergebnis steht unter der Adresse HL.

Die für die Bogen- und Radiusprozedur so wichtige Sinus-Funktion wird über den Einsprung FLOSIN erreicht. Die Routine berechnet von dem durch HL adressierten Wert den Sinus und liefert das Ergebnis unter der Adresse in HL zurück.

Die – ebenso wichtige – Cosinus-Funktion wird über FLOCOS erreicht. Der Cosinus der durch HL adressierten Zahl wird berechnet und das Ergebnis unter der Adresse HL zurückgeliefert.

Die Integer-Routine INTABS liefert in HL den Betrag von HL zurück. Zugleich wird im Register B das ehemalige Vorzeichen zurückgeliefert.

Die Routine ISIGNB liefert den Wert von HL mit dem Vorzeichen von B zurück und arbeitet somit umgekehrt zu INTABS.

INTADD addiert die Werte HL und DE. Das Ergebnis steht in HL, und die Flags werden entsprechend dem Ergebnis gesetzt.

Zur Subtraktion dient die Routine INTSB1, die DE von HL subtrahiert und das Ergebnis in HL zurückgibt.

Ähnlich arbeitet INTSB2, mit dem einzigen Unterschied, daß HL von DE subtrahiert wird.

INTVGL vergleicht HL mit DE. Die im Akku zurückgelieferten Werte und Flagzustände sind die gleichen wie bei FLOVGL.

INTVZW wechselt bei dem Wert in HL das Vorzeichen.

INTSGN gibt das Vorzeichen des HL-Wertes zurück. Ein Sprung zu der Routine liefert Akku gleich 1 und keine Flags gesetzt, wenn HL positiv ist. Der Akku enthält 0, und das Zero-Flag ist gesetzt, wenn HL 0 ist; sollte hingegen HL negativ sein, so ist der Akku nach dem Aufruf &FF, und die Flags für Vorzeichen und Carry sind gesetzt.

Das Label DEGREE stellt lediglich die Adresse eines Flags dar, an dem die Routinen für die Winkelfunktionen erkennen, ob im Gradmaß oder im Bo-

genmaß gerechnet werden soll. Eine 0 in dieser Speicherzelle bedeutet RAD, und ein Wert ungleich 0 bedeutet DEG.

Das in Zeile 10 definierte Label JA repräsentiert nur den Wert &FF, der bei allen zutreffenden Entscheidungen gesetzt wird. Der Wert NEIN wird durch das gleichnamige Label mit dem Wert 0 dargestellt.

Allgemeine Routinen für alle Befehlserweiterungen

Mit der Zeile 230 wird der Anfang des eigentlichen Programms bestimmt. Die ORG-Anweisung setzt den Adressenzähler des Assemblers auf den angegebenen Wert, wodurch der Assembler den erzeugten Maschinencode ab dieser Adresse ablegt.

In den Zeilen 250 bis 300 stehen die Anweisungen für die Einbindung der Befehle in das System.

In der Sprung- und Namenstabelle befinden sich die Assembleranweisungen IF und END. Mit diesen wird eine bedingte Assemblierung erreicht. Steht hinter dem IF-Befehl ein Wert ungleich 0, so wird der Teil bis zum nächsten END assembliert. Ansonsten wird dieser Teil des Quelltextes bei der Assemblierung nicht beachtet. Die Befehle sind auch für die Assemblierung der entsprechenden Labeltabelle in den Zeilen 650 bis 1350 zuständig.

In den Zeilen 310 bis 460 steht die für die Einbindung sehr wichtige Sprungtabelle. Die ersten beiden Bytes der Sprungtabelle müssen die Adresse der Namen darstellen, dies geschieht mit Hilfe der Anweisung DEFW NAMEN. Mit dieser Anweisung erzeugt der Assembler ein Zwei-Byte-Wort, wobei zuerst das niederwertige Byte und dann das höherwertige Byte in den Speicher geschrieben wird. Die DEFW-Anweisung nimmt den dahinterstehenden Integer-Wert, hier das Label NAMEN, und schreibt die diesem Wert entsprechenden Bytes in den Speicher. Auf diese zwei Bytes folgen dann die Sprünge zu den einzelnen Grafik-Routinen.

In der gleichen Reihenfolge wie die Sprünge stehen dann in den Zeilen 470 bis 620 die Namen der Befehle. Jeder Name wird durch den letzten Buchstaben mit gesetztem Bit 7 beendet. Das Ende der Namenstabelle wird durch ein Byte mit dem Wert 0 angegeben.

Nach der Namenstabelle werden mit dem Befehl EXTTAB: DEFS 4 vier Byte für das Betriebssystem reserviert. Dies sind bisher allgemein benötigte Programmteile. Ebenfalls allgemein benötigte Programmteile befinden sich am Ende des Listings und beginnen in Zeile 6370.

DRDX2 ist ein Programmteil, der nur von der VOLLQUADER-Routine genutzt wird und nur ein DRAWR deltax2,0 bewirken soll. Der erste Befehl ist

ein Ladebefehl für das DE-Register mit dem Operanden &0000, so sieht es zumindest der Assembler. Die nächste Zeile ist die Definition eines Labels, dort steht:

```
DX2: EQU $-2
```

Diese Anweisung ist ein Trick, der in den Programmen öfters vorkommt. Das Dollarzeichen ist bei den meisten Assemblern das Symbol für die aktuelle Adresse. Sie bedeutet, daß dem Label DX2 der Wert von $-2, also der aktuellen Speicheradresse-2, zugewiesen wird. Diese Adresse stellt jedoch die Adresse des Operanden des Ladebefehls dar. Dadurch kann man erreichen, daß sich das Programm während der Abarbeitung selbst ändert. Das hat in diesem Fall den Vorteil, daß Speicherplatz gespart wird, da sonst für die Variable DX2 extra zwei Byte Speicherplatz reserviert werden müßten. Außerdem wird die Ausführungszeit für das Programm kürzer, da der Befehl LD DE,(DX2) nicht nur vier Byte Speicherplatz benötigt, sondern auch eine wesentlich längere Ausführungszeit hat. Also noch mal kurz: Die zwei Speicherzellen des Operanden bei diesem Ladebefehl stellen zugleich den Speicherplatz für eine Integer-Variable dar.

Wird nun auf die durch das Label adressierten Speicherzellen zugegriffen und ein Wert abgespeichert, so wird der Operand des Ladebefehls geändert und damit der abgespeicherte Wert von deltax2 direkt geladen.

Der nächste Befehl ist nur dazu da, damit der Y-Wert für den folgenden Ansprung der DRAWR-Routine 0 ist. Der Rücksprung aus der DRDX2-Routine erfolgt über den RET-Befehl in der Betriebssystem-Routine.

Der Programmteil SCHRAE (steht für schräg) in den Zeilen 6420 bis 6460 dient dazu, ein DRAWR mit den unter ALTX und ALTY abgespeicherten Koordinaten durchzuführen. Die Bezeichnungen ALTX und ALTY kommen daher, daß sich die Routine für den Kreis die alten X- und Y-Koordinaten merken muß und dazu diese Speicherzellen benutzt. Hier wird mit den Operanden bei direkten Ladebefehlen der gleiche Trick wie bei der DRDX2-Routine angewandt. Ansonsten stehen ALTX und ALTY für die Adressen der Variablen deltax und deltay, die bei der QUADER- und der VOLLQUADER-Routine Verwendung finden. Vor der Anwendung der Routine werden die zugehörigen Werte für deltax und deltay in die Operandenspeicherzellen abgespeichert. Der Rücksprung erfolgt wieder über den RET der DRAWR-Routine.

Das Unterprogramm QUADRI in den Zeilen 6480 bis 6510 wandelt die in HL übergebene Integer-Zahl in eine durch DE adressierte Fließkommazahl um und quadriert diese anschließend durch eine Multiplikation mit sich selbst.

Das Unterprogramm RECHNE berechnet aus dem durch HL adressierten Winkel und den in RADX und RADY abgespeicherten Radien die X- bzw. Y-Koordinate und speichert sie in den durch X_INT bzw. Y_INT adressierten Speicherzellen ab. Dazu wird zuerst die Adresse des Winkels auf den Stapel gerettet und DE mit der Adresse eines Zwischenspeichers versorgt. Anschließend wird durch den Aufruf von COPYHD der Fließkommawert von der Adresse HL zu der Adresse DE kopiert. Durch einen Austauschbefehl wird HL mit der Adresse des Zwischenspeichers geladen. Mit dieser Adresse wird die Routine FLOSIN aufgerufen, die den dem Winkel entsprechenden Sinus berechnet und bei der Adresse HL speichert. Darauf wird DE mit der Adresse von radiusy geladen und die Routine FLOMUL aufgerufen, worauf der Sinus des Winkels mit radiusy multipliziert wird. Die anschließende Umwandlung zur Integer-Zahl mit FLO_HL liefert eine Information über das Vorzeichen im Akku zurück, die durch ISIGNB an den Integer-Wert übertragen wird. Die so berechnete Y-Koordinate wird bei der Adresse Y_INT abgespeichert.

Die Routine INTFLO dient dazu, Integer-Werte mit Vorzeichen in Fließkommawerte mit Vorzeichen umzuwandeln. Dazu wird zuerst die Zieladresse des Fließkommawertes gerettet und der Betrag des Integer-Wertes gebildet. Die Adresse des Fließkommaspeichers wird wieder geholt und der Integer-Wert mit der HL_FLO-Routine in einen Fließkommawert umgewandelt. Jetzt wird das von der Konvertierungs-Routine unbeeinflußte B-Register, das das ehemalige Vorzeichen des Integer-Wertes enthält, in den Akku geladen. Dort wird das Vorzeichen geprüft und bei Negativität ein Vorzeichenwechsel des Fließkommawertes mit für die Parameterübergabe geretteten IX-Register durchgeführt. Sonst erfolgt ein ganz normaler Rücksprung.

Das Unterprogramm MITTE dient dazu, mit den gewünschten Mittelpunktskoordinaten und den alten Koordinaten des Grafikursprungs die neuen Koordinaten für den Grafikursprung zu berechnen und den Ursprung zu setzen. Die neuen X- und Y-Koordinaten des Mittelpunkts werden in DE und HL übergeben. Die Routine rettet zuerst den neuen Y-Wert, also das HL-Register, auf den Stapel. Dann wird das HL-Register mit der alten X-Koordinate des Ursprungs geladen und der neue X-Wert im DE-Register dazu addiert. Die neue X-Koordinate für den Ursprung wird mit einem Austauschbefehl auf den Stapel gerettet und zugleich der neue Y-Wert wieder vom Stapel geholt. Jetzt wird die alte Y-Koordinate ins DE-Register geladen und zum Y-Wert des gewünschten Mittelpunkts addiert. Die neue Y-Koordinate für den Ursprung steht nun im HL-Register, und die neue X-Koordinate wird vom Stapel ins DE-Register geholt. Mit einem Sprung zum Ursprung-Setzen wird der Mittelpunkt festgelegt. Der Rücksprung erfolgt über den RET der Ursprung-Routine.

Das nächste Unterprogramm heißt RAD und stellt das Flag für Bogen- oder Gradmaß auf Bogenmaß um. Zuvor wird allerdings der alte Zustand des Flags unter der Adresse FLAG abgespeichert. Um die Winkelfunktions-Routinen auf Bogenmaß umzustellen, muß die Speicherzelle mit dem Wert 0 geladen werden. Der Akku wird dazu mit XOR A gelöscht und in der Speicherzelle mit der Adresse DEGREE abgespeichert. Zum Schluß erfolgt der Rücksprung zur aufrufenden Routine.

Die Routine DEG arbeitet wie RAD, mit dem einzigen Unterschied, daß das Flag DEGREE mit &FF geladen wird, damit alle Winkelfunktionsberechnungen im Gradmaß erfolgen.

Das Unterprogramm DEGRES – der Name steht für DEG RESTORE – stellt den ursprünglichen Zustand des Flags für die Berechnung im Grad- oder Bogenmaß wieder her. Der alte Zustand des Flags wurde dazu von den Routinen DEG und RAD in das Operandenbyte des LD A,nn-Befehls abgespeichert. Dadurch wird wieder Speicherplatz gespart, da das alte Flag nicht in einem extra reservierten Byte untergebracht werden muß. Außerdem benötigt der LD A,nn-Befehl nur zwei Byte Maschinencode, während der LD A,(FLAG)-Befehl drei Byte für die Ausführung benötigt. Wie üblich erfolgt der Rücksprung zum Schluß der kurzen Routine.

GRSTO ist ein Unterprogramm, das von jeder Grafik-Routine aufgerufen wird. Es speichert die Position des Grafik-Cursors sowie des Koordinatenursprungs und die Grafikstiftfarbe. Die Werte werden in die Operanden der Routine GRREST transferiert.

GRREST ist die Routine, die den Eintrittszustand des Grafiksystems wieder herstellt. Dazu wird die Farbe des Grafikstifts in den Akku geladen und damit der Grafikstift gesetzt. Dann wird die X-Koordinate des Ursprungs ins DE-Register und die Y-Koordinate ins HL-Register geladen und mit SETORG der Ursprung wieder hergestellt. Als letztes werden die beiden Registerpaare DE und HL mit der X- bzw. Y-Koordinate des ursprünglichen Grafik-Cursors geladen, und der Grafikcursor wird mit der Routine MOVE wieder auf den Eintrittswert gesetzt. Der Rücksprung erfolgt über die Routine MOVE.

Das Unterprogramm VERGL ist eine Vergleichsroutine, die die Werte von DE und HL vergleicht, in HL den größeren und in DE den kleineren Integer-Wert zurückliefert. Als Vergleichsroutine wird INTVGL eingesetzt, und wenn HL größer ist, wird die Routine verlassen. Sonst werden die beiden Registerinhalte vertauscht, und dann wird die Routine verlassen.

Die Routine COPYHD in den Zeilen 7430 bis 7510 kopiert die fünf Bytes einer Fließkommavariablen von der Adresse HL an die Adresse DE. Die Inhalte der Register werden bis auf das Parity/Overflow-Flag nicht geändert. Dazu

werden die Registerpaare BC, DE und HL auf den Stapel gerettet. Das BC-Register wird mit fünf, also der Anzahl der Bytes, geladen und dann der Blockverschiebebefehl LDIR durchgeführt. Vor dem Rücksprung erfolgt zum Schluß das Holen der Registerpaare HL, DE und BC vom Stapel.

Die Routinen in den Zeilen 7530 bis 7790 laden abwechselnd die Registerpaare HL und DE indiziert mit dem IX-Register und einem Offset. Diese Routinen werden immer dann benötigt, wenn die übergebenen Parameter oder deren Adressen geholt werden müssen. Bei einem CALL oder einer Befehlserweiterung werden die Parameter auf dem Stapel abgelegt und durch das IX-Register adressiert. Dabei zeigt das IX-Register auf den zuletzt übergebenen Parameter. Alle Parameter, die vor dem letzten kamen, werden mit jeweils einem Offset von 2 adressiert. Wenn also n Parameter übergeben wurden, so erreicht man den i-ten unter der Adresse IX+(2*(n−i)).

Die letzten acht Zeilen ab der Zeile 7810 stellen eine Fließkommakonstante mit dem Wert 1 und den Speicherplatz für sieben Fließkommavariablen dar. Unter Y^2 wird das aktuelle Quadrat des Y-Wertes abgespeichert und zur Berechnung des X-Wertes in den Routinen KREIS und SCHEIBe genutzt. RADX und RADY stellen den Speicher für die Radien bzw. der Quadrate derselben. Unter der Adresse WINKEL wird der aktuelle Winkel, und damit am Anfang der Anfangswinkel, der BOGEN- und RADIUS-Routinen gespeichert. ZWIE stellt den Zwischenspeicher für eine Fließkommazahl dar. Unter WIEND wird der Endwinkel der Bogen-Routinen gespeichert und schließlich unter WISTEP der Winkel, um den der aktuelle Winkel jeweils erhöht werden soll.

Soviel zu den allgemein genutzten Routinen und den Speichern für die Fließkommazahlen. Es gibt noch ein paar Routinen, die von mehreren Grafikbefehlen benutzt werden. Diese werden aber während der Programmerläuterung der einzelnen Befehle angesprochen.

RECHTECK-Befehl

Aufgerufen wird dieser Befehl von dem JP RE_ECK in der Zeile 320. Die Routine beginnt in Zeile 1380 und prüft dort zuerst den Akkuinhalt auf 5, um festzustellen, ob tatsächlich alle Parameter übergeben wurden. Ist der Akkuinhalt ungleich 5, so kehrt die Routine sofort ins BASIC zurück. Wenn die richtige Anzahl von Parametern übergeben wurde, werden zuerst mit dem Aufruf von GRSTO die Zustände der Grafik abgespeichert.

Dann wird die Routine FARBE aufgerufen. Dies ist die gleiche Routine wie GRPEN, mit dem Unterschied, daß nicht mehr die Parameterzahl überprüft

wird. Durch diese Routine wird der Grafikstift auf den letzten Parameter gesetzt.

Als nächstes wird das HL-Register von der Adresse IX+6 geladen, also der Parameter y1 geholt. Dieser Y-Wert wird in der Adresse Y_INT abgespeichert. Darauf wird der Parameter x1 mit der Routine GTDE8X von der Adresse IX+8 ins DE-Register geladen und unter der Adresse X_INT abgespeichert. Für ein Rechteck braucht man je zwei X- und Y-Werte, deshalb werden jetzt mit den Routinen GTDE4X und GTHL2X die Parameter x2 und y2 in die Register DE und HL geladen.

Nachdem alle X- und Y-Koordinaten geholt wurden, wird das Unterprogramm R_ECK aufgerufen. Es zeichnet ein Rechteck mit den Koordinaten DE, HL, X_INT und Y_INT.

Die Rechteck-Routine wird durch den Sprung zur GRREST-Routine beendet, in der die Grafikzustände wieder hergestellt werden.

Das Unterprogramm R_ECK beginnt in der Zeile 1510 und wird von den Grafik-Routinen RECHTECK und QUADER benötigt. Die Routine benötigt zwei Koordinatenpaare. Das eine muß an die Adressen X_INT und Y_INT geladen und das andere in DE und HL übergeben werden. Die beiden Koordinaten in DE und HL werden zuerst auf den Stapel gerettet. Dann wird, weil die MOVE-Routine die Registerinhalte zerstört, das DE-Register nochmals auf den Stapel gerettet und die Routine MOVE aufgerufen, die ein MOVE zu dem Punkt (DE,HL) durchführt. Das DE-Register wird nach Ausführung der Routine wieder geholt, und HL wird mit dem Wert von Y_INT geladen. Jetzt wird das HL-Register auf den Stapel gerettet und mit dem Aufruf von DRAW eine Linie zum Punkt (DE,Y_INT) gezogen. HL wird vom Stapel geholt, DE mit dem Wert von X_INT geladen und auf den Stapel gerettet. Der Aufruf von DRAW zieht eine Linie zum Punkt (X_INT,Y_INT). Das DE-Register wird vom Stapel geholt, ebenso das zu Anfang gerettete HL-Register. HL wird nochmals gerettet, und mit DRAW wird eine Linie zum Punkt (X_INT,HL) gezogen. HL wird ein letztes Mal geholt und das zu Anfang gerettete DE-Register ebenfalls. Mit einem letzten DRAW wird eine Linie zum Punkt (DE,HL) gezeichnet. Damit ist die Routine am Ausgangspunkt angelangt und kehrt mit dem Rücksprung der DRAW-Routine zurück.

BLOCK-Befehl

Die Routine wird von Zeile 330 initiiert und steht in den Zeilen 1730 bis 2060. Zuerst wird wieder die Parameterzahl überprüft und bei einer anderen als 5 ins BASIC zurückgekehrt.

Ansonsten werden mit der Routine GRSTO die Zustände des Grafiksystems abgespeichert, und die Grafikstiftfarbe wird mittels der Routine FARBE eingestellt. Als nächstes werden die beiden X-Koordinaten x1 und x2 geholt und an den Adressen X1 und X2 abgespeichert. An diesen Adressen stehen sie als Operand von LD DE,nnnn-Befehlen im Programm zur Verfügung.

Der folgende Programmschritt holt die beiden Y-Koordinaten in die Register DE und HL. Diese werden mit der Routine VERGL verglichen, und der größere Wert wird im HL- bzw. der kleinere im DE-Register zurückgegeben. Der größere Wert, also das HL-Register, wird dann bei der Adresse Y_INT gespeichert.

Mit den Y-Werten wird nun eine Schleife betreten. In ihr wird zuerst das HL-Register aus der Adresse Y_INT geladen und der kleinere – auf einen geraden Bildpunkt korrigierte – Y-Wert auf den Stapel gerettet. Die Adresse Y_INT stellt den Zwischenspeicher für den laufenden Y-Wert dar.

Jetzt lädt die Routine das DE-Register mit dem Wert x1 und rettet den Y-Wert in HL auf den Stapel. Mit x1 in DE und y in HL wird der Grafik-Cursor mit der MOVE-Routine auf den Punkt (x1,y) gesetzt.

Danach holt ein Befehl den Y-Wert wieder vom Stapel und ein anderer lädt anschließend das DE-Register mit dem Wert x2. Der Y-Wert wird wieder gerettet. Mit diesen Werten zieht die DRAW-Routine eine horizontale Linie zum Punkt (x2,y).

Der Y-Wert wird anschließend wieder vom Stapel geholt und zweimal dekrementiert, also um 2 erniedrigt. Diese Erniedrigung um 2 macht eine Routine schneller, da von den Y-Koordinaten je zwei denselben Bildschirmpunkt ansteuern.

Richtet man in einer Grafik-Routine eine Schleife ein, die die Y-Werte durchläuft, so ist diese Routine aus zwei Gründen besser als eine, die die X-Werte dazu hernimmt. Erstens ist der Unterschied zwischen zwei Punkten in Y-Richtung immer gleich, wohingegen der Unterschied in X-Richtung vom Modus abhängt, und zweitens werden bei einer horizontalen Linie immer benachbarte Bytes beschrieben, während bei vertikalen Linien immer nur ein paar Bits aus einem Byte selektiert und beschrieben werden müssen.

Nach der Erniedrigung des Y-Wertes wird er wider bei der Adresse Y_INT abgespeichert. Jetzt wird der niedrigere Y-Wert vom Stapel nach der Löschung des Carry-Flags vom laufenden Y-Wert subtrahiert. Solange das Ergebnis dieser Subtraktion positiv ist, also der laufende Y-Wert größer als der kleinere Y-Wert ist, wird zum Schleifenanfang gesprungen und die nächste Linie gezeichnet.

Ansonsten kehrt die Routine über die Routine GRREST, die die Werte des Grafiksystems wieder herstellt, zum BASIC zurück.

KREIS-Befehl und SCHEIBE-Befehl

Diese beiden Befehle werden durch ein und dieselbe Routine implementiert. Nur die Auswertung der berechneten Ergebnisse ist unterschiedlich. Sie geschieht durch eine entsprechende Zeichen-Routine.

Die Routine KREIS prüft die Anzahl der Parameter. Ist diese ungleich 5, kehrt sie ins BASIC zurück. Dann lädt sie den Akku mit &FF und speichert diesen Wert unter der Adresse FLAG ab. Diese Speicherzelle gibt der Zeichen-Routine dann an, ob ein Ellipsenrahmen oder eine ausgefüllte Ellipse zu zeichnen ist.

Dann springt die Routine zum Weitermachen an die Stelle hinter der Festlegung der SCHEIBE. Die Routine SCHEIBe prüft ebenfalls die Parameterzahl und kehrt bei einem Wert ungleich 5 ins BASIC zurück. Von dieser Routine wird allerdings der Akku gelöscht und dieser Wert, also 0, unter der Adresse FLAG gespeichert. An der gelöschten Speicherzelle erkennt die Routine ZEICHNen, daß die Ellipse ausgefüllt werden muß.

Ab jetzt geht es für beide Routinen identisch weiter. Zuerst werden die Grafikzustände gespeichert, und die Farbe des Grafikstifts wird hergestellt.

Als zweites werden die Werte mittex und mittey in die Registerpaare DE und HL geladen, und damit wird die Routine MITTE aufgerufen. Diese setzt den Grafikursprung als Mittelpunkt, wobei die ursprünglichen Ursprungskoordinaten berücksichtigt werden.

Als nächstes wird HL mit 0 geladen und an der Adresse ALTX abgelegt, damit die Variable altx einen definierten Anfangswert hat.

Dann wird das IX-Register auf den Stapel gerettet, da es von der Arithmetik-Routine QUADRI verändert wird. Darauf wird radiusx ins DE-Register und HL mit der Adresse RADX geladen, bevor die Inhalte vertauscht werden. Mit diesen Werten wird nun die Routine QUADRI aufgerufen, die den Integer-Wert von HL in eine Fließkommazahl an der Adresse DE umwandelt und daraufhin mit sich selbst multipliziert, also quadriert. Nach der Quadrierung von radiusx wird das IX-Register wieder vom Stapel geholt. Mit dessen Hilfe wird der Wert radiusy in das HL-Register geladen, das sofort noch auf den Stapel gerettet wird. In das DE-Register wird die Adresse RADY geladen und die Routine QUADRI wieder aufgerufen. Diesmal quadriert sie radiusy und schreibt es an die Adresse RADY.

Der Wert von radiusy stellt zugleich den Anfangswert für eine Schleife mit Y-Werten dar. Dazu wird radiusy vom Stapel in HL geholt und davon erst einmal der Betrag gebildet. Dann wird der Wert noch auf einen geraden Anfangswert gebracht. Dies geschieht, da jeweils eine gerade und die nächsthöhere ungerade Y-Koordinate den gleichen Pixel ansprechen. Wenn nun die Y-Koordinate im Schleifendurchlauf um jeweils 2 erniedrigt wird, so kann man sehr einfach den letzten Wert prüfen, er müßte 0 sein.

Einen geraden Wert erhält man dadurch, daß man das niederwertigste Bit auf 0 setzt. Die so entstandene Koordinate stellt den Anfangswert für alty dar, der – ebenso wie altx – nur für den Ellipsenbogen von Interesse ist.

Mit dem Y-Wert wird jetzt eine Schleife namens Y_LOOP betreten. In dieser Schleife wird zuerst der Y-Wert unter der Adresse Y_INT gespeichert. Dann wird DE mit der Adresse Y^2 geladen und der Y-Wert dorthin quadriert. Das HL-Register enthält darauf die Adresse Y^2. DE wird nun mit der Adresse RADY geladen und mit dem Aufruf von FLODIV, y^2 durch radiusy^2 dividiert. HL enthält die Adresse des Ergebnisses (Y^2). DE wird nun mit der Adresse der Fließkommakonstanten 1 geladen, und mit der Routine FLOSB2 wird das vorherige Ergebnis von 1 subtrahiert. Jetzt wird DE mit RADX geladen, also der Adresse des quadrierten radiusx, und mittels FLOMUL mit der zuvor erhaltenen Differenz multipliziert. Von diesem Ergebnis wird dann noch die Quadratwurzel berechnet, das Ergebnis in eine Integer-Zahl umgewandelt und als Koordinate unter der Adresse X_INT gespeichert. Für diejenigen, denen dieser Vorgang zu unübersichtlich war, soll die Berechnungsformel in Kurzform aufgezeigt werden:

x = SQR (radiusx^2 * (1 – (y^2 / radiusy^2)))

Formt man diesen Ausdruck um, so kommt man auf folgende Formel:

(x^2 / radiusx^2) + (y^2 / radiusy^2) = 1

Jetzt werden manche unter Ihnen eine bekannte Gleichung lesen, denn dies ist die Gleichung der Normalform einer Ellipse. Diese Ellipse hat die Radien radiusx und radiusy und hat den Mittelpunkt auf dem Ursprung. Das ist möglich, weil die Routinen für Ellipse, Scheibe und Bogen den Ursprung auf den Mittelpunkt verlegen und damit die Berechnung vereinfachen bzw. den Grafik-Routinen überlassen.

Es wurden also die Koordinaten unter den Adressen Y_INT und X_INT gespeichert. Mit diesen Werten wird die Routine ZEICHNen angesprungen.

Nach der Zeichen-Routine wird HL mit dem Y-Wert geladen. Dann wird geprüft, ob der Y-Wert bereits 0 ist, und anschließend um 2 vermindert. War der

Y-Wert ungleich 0, so springt die Routine wieder an den Schleifenanfang von Y_LOOP. Ansonsten wird die Routine über die GRREST-Routine verlassen.

Wichtig für die KREIS- und SCHEIBEn-Routine ist jedoch das eben erwähnte Unterprogramm ZEICHNen, das die Werte von ALTX, ALTY, X_INT und Y_INT benutzt. Diese Werte werden in Abhängigkeit von dem Flag für den leeren Kreis benutzt.

Zuerst werden die soeben berechneten Werte von X_INT und Y_INT in die Register DE und HL geladen. Diese Werte werden dann auf den Stapel gerettet, und mit ihnen wird die PLOT-Routine aufgerufen. Es erfolgt also ein PLOT x,y.

Nach dem PLOT wird das Flag geprüft und bei 0 zum Label VOLL1 gesprungen. Ansonsten werden DE und HL mit altx und alty geladen. HL und DE werden auf den Stapel gerettet, und die DRAW-Routine wird aufgerufen. Die Routine führt also ein DRAW altx,alty aus. Dann wird altx vom Stapel ins HL-Register geholt und mit der INTVZW-Routine das Vorzeichen gewechselt. DE wird vom Stapel geholt und enthält somit alty. Um die Koordinaten im richtigen Register zu haben, werden nun die Registerinhalte vertauscht. Der folgende Befehl PLOT −altx,alty wird noch durchgeführt, bevor mit dem Label VOLL1 mit dem für SCHEIBE und KREIS gemeinsamen Teil weitergemacht wird.

Als nächstes wird x vom Stapel ins HL-Register geholt und mit INTVZW das Vorzeichen gewechselt. Das DE-Register wird mit y vom Stapel geladen und, um −x in DE und y in HL zu haben, mit HL vertauscht. Der Aufruf der DRAW-Routine bewirkt ein DRAW −x,y.

Jetzt wird HL aus Y_INT geladen und mit INTVZW negiert. Danach wird DE aus X_INT geladen, und beide Register werden auf den Stapel gerettet. Nach dem Retten von x und −y wird mit diesen Werten die PLOT-Routine aufgerufen, die den Grafik-Cursor auf den Punkt (x,−y) setzt. Anschließend wird wieder das Flag für den leeren Kreis getestet und bei 0 zum Label VOLL2 verzweigt.

Für den leeren Kreis geht es damit weiter, daß HL mit alty geladen und negiert wird. Daraufhin wird DE mit altx geladen, und beide Werte werden auf den Stapel gerettet. Der folgende Befehl zieht eine Linie mit DRAW altx,−alty. Darauf wird altx vom Stapel ins HL-Register geladen und negiert. DE wird vom Stapel mit −alty geladen, und beide Registerpaare werden vertauscht. Ein anschließendes PLOT geht an die Koordinaten (−altx,−alty). Die Routine für die Kreislinie überspringt jetzt einen Prüfungsabschnitt der Scheiben-Routine.

Für die Scheiben-Routine geht es damit weiter, daß HL von Y_INT geladen und auf 0 geprüft wird. Dies geschieht deshalb, damit die letzte Linie, also die durch den Mittelpunkt, nicht zweimal gezeichnet wird. Das ist wichtig, wenn der Grafikverknüpfungsmodus 1 – XOR-Verknüpfung – gewählt wird. In diesem Fall würde nämlich dann die mittlere Linie immer gelöscht werden. Ist HL also 0, so wird der Stapel gereinigt und zur Hauptroutine zurückgekehrt; wenn nicht, wird bei dem Label K_LEER für Scheibe- und Kreis-Routine weitergemacht.

Dort wird x vom Stapel ins HL-Register geladen und negiert. Darauf kommt −y vom Stapel ins DE-Register, und die Register werden vertauscht. Der Aufruf von DRAW zieht eine letzte Linie zum Punkt (−x,−y). Dann wird ALTY mit Y_INT und ALTX mit X_INT geladen und zur Hauptroutine zurückgekehrt.

Die ZEICHN-Routine führt also, in Abhängigkeit von der zu zeichnenden Ellipse, also ob ausgefüllt oder nicht, unterschiedliche Zeichenbefehle durch. Die Folge der Zeichenbefehle ist nachfolgend dargestellt.

Zeichenbefehle für die verschiedenen Formen:

ausgefüllte Ellipse	unausgefüllte Ellipse
PLOT x, y	PLOT x, y
	DRAW altx, alty
	PLOT −altx, alty
DRAW −x, y	DRAW −x, y
PLOT x, −y	PLOT x, −y
	DRAW altx, −alty
	PLOT −altx, −alty
DRAW −x, −y	DRAW −x, −y

Man sieht hier, daß nur bei der Routine für die leere Ellipse ein paar zusätzliche Befehle nötig sind, um gegenüber der Routine der ausgefüllten Ellipse nur die Ellipsenlinie zu zeichnen. Dies wird bei den Routinen ausgenutzt.

BOGEN-Befehle

Die BOGEN-Routinen beginnen wie alle anderen mit der Prüfung der Parameterzahl und kehren, wenn diese ungleich 8 ist, zum BASIC zurück.

Von der Routine BOGEN wird wistep ins HL-Register geholt, DE mit der Adresse WISTEP geladen und wistep in eine Fließkommazahl umgewandelt. Dann wird DE mit wiend geladen, HL mit der Adresse WIEND, die Register

werden vertauscht, und wiend wird in eine Fließkommazahl umgewandelt. Jetzt wird HL mit wianf geladen, DE mit der Adresse WINKEL und wianf in eine Fließkommazahl umgewandelt. Als letztes wird das Flag für die Winkelfunktionen auf Gradmaß gestellt und zum Label BOCONT zum Weitermachen gesprungen.

BOGEN.D hingegen stellt das Flag der Winkelfunktionen auf Gradmaß und springt zum Label BDCONT.

BOGEN.R stellt das Winkelfunktionsflag auf Bogenmaß.

Ab jetzt geht es für BOGEN.D und BOGEN.R gemeinsam weiter. Es werden nacheinander die Adressen der im BASIC übergebenen Fließkommavariablen geholt und mit COPYHD an die Adressen für die Routinen kopiert. Zuerst wird dazu HL mit der Adresse der BASIC-Variablen wistep und DE mit der Adresse WISTEP geladen und wistep dorthin kopiert. Dann wird DE mit der Adresse von wiend und HL mit WIEND geladen, und nach dem Vertauschen der Register wird die Variable kopiert. Als letztes wird dann DE mit der Adresse WINKEL und HL mit der Adresse der Variablen wianf geladen und mit COPYHD die Variable kopiert.

Jetzt geht die Routine für alle drei Einsprünge gleich weiter. Zuerst werden die Grafikzustände gespeichert, und der Grafikstift wird auf die angegebene Farbe gesetzt. Dann werden die Werte mittex und mittey geholt, und mit der MITTE-Routine wird der Mittelpunkt für den Bogen gesetzt. Anschließend wird radiusy ins DE-Register geholt, HL mit der Adresse RADY geladen und, nach dem Vertauschen der Register, mit der INTFLO-Routine in eine Fließkommazahl umgewandelt. Nun wird radiusx ins DE-Register geholt, HL mit der Adresse RADX geladen und radiusx in eine Fließkommazahl umgewandelt. Jetzt wird noch der Winkelstep auf Negativität oder 0 überprüft, bei Zutreffen einer dieser Bedingungen wird die Routine mit einem Sprung zu BOENDE beendet. Bevor es in die Winkelschleife geht, wird noch das Flag für den ersten Durchgang geladen und HL mit der Adresse des Winkels – WINKEL – versehen. Die Schleife beginnt mit dem Label WILOOP, wobei zuerst die Winkeladresse auf den Stapel gerettet und damit die Routine RECHNE aufgerufen wird. Die Routine gibt unter den Adressen X_INT und Y_INT die berechneten Koordinaten zurück.

Die beiden Koordinaten werden in die Register DE und HL geladen. Der Akku wird mit dem Flag für den ersten Durchgang geladen und auf einen Wert gleich 0 geprüft. Wenn das Flag 0 ist, also nicht der erste Durchgang der Schleife erfolgt, wird bei dem Label WEITE1 weitergemacht. Ansonsten setzt die Routine nur den Grafik-Cursor mit MOVE auf den ersten Punkt. Das Flag wird gelöscht und die DRAW-Routine übersprungen.

Ist das Flag gleich 0, so findet ein DRAW zu dem in DE und HL übergebenen Punkt statt.

Weiter geht es dann damit, daß HL mit der Adresse des aktuellen Winkels (WINKEL) und DE mit der Adresse von wistep (WISTEP) geladen wird. Die beiden Werte werden mit FLOADD addiert. Dann wird DE mit WIEND – der Adresse von wiend – geladen und nach Vertausch von DE und HL ein Vergleich mit FLOVGL durchgeführt. Ist wiend größer oder gleich als der aktuelle Winkel, so geht es, nach dem Zurückholen der Adresse des aktuellen Winkels vom Stapel, am Schleifenanfang weiter.

Ansonsten wird, nach der Wiederherstellung der Grafikzustände mit GRREST, über die Wiederherstellung des Flags der Winkelfunktionen zum BASIC zurückgekehrt.

RADIUS-Befehl

Er beginnt wie alle anderen Befehle mit der Prüfung der Parameterzahl. Ist diese ungleich 6, wird ins BASIC zurückgekehrt.

Ansonsten wird Gradmaß für die Berechnung gesetzt, die Grafikzustände werden gespeichert, und die Grafikstiftfarbe wird gesetzt. Jetzt wird DE mit mittey und HL mit mittex geladen und nach dem Vertauschen der Register der Mittelpunkt mit MITTE gesetzt.

Dann wird radiusx ins HL-Register geholt, DE mit der Adresse RADX geladen und radiusx in eine Fließkommazahl umgewandelt. Nun wird radiusy ins DE-Register geholt, HL mit der Adresse RADY geladen, die Register werden vertauscht, und radiusy wird umgewandelt. Im folgenden wird noch der Winkel ins HL-Register geholt, DE mit der Adresse WINKEL geladen und der Winkel in eine Fließkommazahl umgewandelt.

Mit der – von INTFLO stammenden – Adresse WINKEL in HL wird die Routine RECHNE aufgerufen.

Dann wird DE mit der X- und HL mit der Y-Koordinate geladen und vom Ursprung aus ein DRAW zu dem Punkt durchgeführt. Die Linie wird deshalb vom Ursprung aus gezogen, da die Routine MITTE den Ursprung mit SETORG setzt. Dieses Setzen des Ursprungs setzt immer den Grafik-Cursor zum Ursprung.

Anschließend wird die RADIUS-Routine nach der Herstellung der Grafikwerte und des Zustands des Flags der Winkelfunktionen verlassen und zum BASIC zurückgekehrt.

GRPEN-Befehl

Zuerst wird die Übergabe eines Parameters geprüft und bei falscher Parameterzahl zum BASIC zurückgekehrt. Der Befehl GRPEN ist nur vorhanden, wenn für einen CPC 464 assembliert wurde.

Ansonsten wird der Akku mit dem letzten Parameter geladen und SGRPEN aufgerufen. Dort wird der Grafikstift gesetzt und über den RET-Befehl von SGRPEN zurückgekehrt. Der Teil beim Label FARBE, ab der Parameterprüfung, wird von den meisten Routinen zum Setzen des Grafikstifts aufgerufen.

GRPAPER-Befehl

Der Akku und damit die Anzahl der Parameter wird auf 1 geprüft, und bei Ungleichheit wird ins BASIC zurückgekehrt. Ansonsten wird der Akku mit der gewünschten Farbe geladen und das Grafikpaper mit SGRPAP gesetzt. Die Rückkehr zum BASIC erfolgt über die SGRPAP-Routine. GRPAPER wird nur assembliert, wenn das Label CPC 464 mit JA belegt wurde.

GRMODE-Befehl

Es wird wieder auf einen Parameter geprüft und sonst zum BASIC zurückgekehrt. Dann wird der Akku mit dem letzten Parameter geladen und mit dem Sprung zu SGRMOD der Grafikmodus gesetzt und ins BASIC zurückgekehrt.

Besitzer der Versionen 664 und 6128 benötigen keinen GRMODE-Befehl, da sie die XOR-Darstellung als Farbstift-Modus angeben können. Da dies nicht nur bei den Befehlen DRAW, DRAWR, PLOT und PLOTR, sondern auch bei MOVE sowie MOVER erfolgen kann, ist eine Verbindung zu den neuen Befehlen gegeben.

QUADER-Befehl

Bei einer Parameterzahl verschieden von 7 kehrt die Routine ins BASIC zurück. Sonst werden die Grafikzustände gespeichert, und der Grafikstift wird gesetzt.

Als nächstes wird DE mit x1 geladen und unter der Adresse X_INT gespeichert. HL wird darauf mit y1 geladen und unter Y_INT abgespeichert. Dann wird x2 ins DE- und y2 ins HL-Register geladen, und beide Werte werden auf den Stapel gerettet. Mit den vier Koordinaten wird nun R_ECK aufgerufen, um das vordere Rechteck des Quaders zu zeichnen.

Nach dem Rechteck wird y2 vom Stapel ins DE-Register und HL von der

Adresse Y_INT geladen. Die beiden Y-Werte werden verglichen, und der größere wird unter OBENY gespeichert und auf den Stapel gerettet. Der kleinere wird bei der Adresse UNTENY gespeichert.

Hierauf wird untenhinteny ins HL-Register geholt und unteny davon subtrahiert. Der entstandene Wert ist deltay und wird für die SCHRAEg-Routine unter ALTY gespeichert. Obeny wird vom Stapel ins DE-Register geladen und zum Wert deltay addiert. Der entstandene Wert obenhinteny wird unter Y_INT gespeichert.

Jetzt wird x2 vom Stapel ins DE-Register geholt und HL aus der Adresse X_INT mit x1 geladen. Die beiden Werte werden mit VERGL verglichen, der größere Wert wird unter RECHTX gespeichert und auf den Stapel gerettet. Der kleinere Wert wird unter LINKSX gespeichert und mit dem Vertauschbefehl ins HL-Register transferiert. Linkshintenx wird in DE geholt, und linksx wird davon abgezogen. Der entstandene Wert deltax wird unter ALTX gespeichert. Rechtsx wird vom Stapel in DE geholt und zu deltax addiert, so daß rechtshintenx entsteht und unter X_INT gespeichert werden kann.

Zu obenhinteny und rechtshintenx werden DE und HL noch mit linkshintenx und untenhinteny geladen, und die Routine R_ECK wird aufgerufen, die daraufhin das hintere Rechteck zeichnet.

Jetzt müssen nur noch die vier schrägen Verbindungslinien gezogen werden. Dazu wird HL mit obeny und DE mit rechtsx geladen, und beide werden auf den Stapel gerettet. Ein folgender MOVE-Befehl setzt den Grafik-Cursor auf die rechte obere Ecke des vorderen Rechtecks. Von dort wird mit SCHRAE die erste Verbindungslinie gezeichnet. DE wird vom Stapel mit rechtsx und HL aus UNTENY geladen. Unteny wird auf den Stapel gerettet. Es erfolgt ein MOVE rechtsx,unteny und das Zeichnen der zweiten schrägen Linie. Unteny wird vom Stapel in HL geholt und DE von LINKSX geladen und auf den Stapel gerettet. Es erfolgt ein MOVE linksx,unteny und die dritte Verbindungslinie. Zur letzten Verbindungslinie wird linksx vom Stapel in DE und obeny in HL geholt und MOVE aufgerufen. SCHRAE zeichnet die letzte Verbindungslinie vom Punkt (linksx,obeny) aus. GRREST stellt den Zustand der Grafik wieder her, und die QUADER-Routine wird verlassen.

VOLLQUADER-Befehl

Die Routine prüft zuerst auf 9 Argumente und kehrt bei anderer Anzahl ins BASIC zurück. Liegen 9 Argumente vor, wird zunächst der letzte Parameter geholt und unter FARBE3 abgespeichert. Dann wird der vorletzte geholt und unter FARBE2 gespeichert, und schließlich wird der drittletzte unter FARBE1 gespeichert. Diese Abspeicherung dient dazu, daß während des Abarbei-

tens des Programms die Farben für die jeweiligen Flächen schnell eingestellt werden können, da sie als Operanden von LD A,nn-Befehlen vorliegen.

Jetzt werden die Grafikzustände mit GRSTO gespeichert. Die Zeilen 5160 bis 5220 dienen dazu, den vom Bildschirmmodus abhängigen Step in X-Richtung zu berechnen und abzuspeichern. Dazu wird GTMODE aufgerufen, wodurch im Akku der aktuelle Modus zurückgegeben wird. Der Modus wird um 1 erhöht und ins Register 8 geschrieben, der Akku wird mit 8 geladen. Das geschieht, weil die folgende Schleife mindestens einmal durchlaufen wird. Dabei wird der Akkuinhalt immer um eine Stelle nach rechts rotiert (durch 2 dividiert), da an der linken Seite keine gesetzten Bits hereinrotiert werden können. Bei MODE 0 enthält der Akku nach der Schleife den Wert 4, bei MODE 1 den Wert 2 und bei MODE 2 den Wert 1. Dies sind genau die für die X-Werte nötigen Schrittweiten der Grafik-Routinen. Diese Schrittzahl wird unter der Adresse STEP gespeichert.

Jetzt wird x1 in DE und x2 in HL geholt, und die beiden Werte werden mit VERGL verglichen. Der größere wird unter RECHTSX und der kleinere unter LINKSX gespeichert. Außerdem wird der größere auf den Stapel gerettet und der kleinere ins HL-Register gewechselt.

DE wird nun mit linkshintenx geladen, und linksx – in HL – wird davon subtrahiert. Der entstandene Wert deltax wird unter ALTX – für die SCHRAE-Routine – gespeichert. Das Vorzeichen von deltax wird getestet, und der oberste Wert auf dem Stapel wird mit deltax ausgetauscht. Es kommt also deltax auf den Stapel, während rechtsx geholt wird.

War das Vorzeichen positiv, so wird unter X_INT linksx gespeichert und die Differenz rechtsx–linksx berechnet. Ansonsten wird rechtsx unter X_INT gespeichert und (linksx–rechtsx) berechnet. Die berechnete Differenz wird als deltax2 unter DX2 gespeichert.

Nun wird HL mit y2 geladen und nach DE transferiert, damit y1 ins HL-Register geholt werden kann. Die beiden Register werden verglichen, der größere Wert unter OBENY und der kleinere unter UNTENY gespeichert. Zudem wird der größere auf den Stapel gerettet und untenhinteny ins HL-Register geholt. Der Wert deltay ergibt sich aus der Differenz (untenhinteny–unteny) und wird – für die Routine SCHRAE – unter ALTY gespeichert. Daraufhin wird das Vorzeichen von deltay geprüft und obeny vom Stapel in HL geholt.

Ist deltay größer 0, so wird obeny nach Y_INT gespeichert und unteny–obeny berechnet. Ist deltay jedoch negativ, so wird unteny unter Y_INT gespeichert und die Differenz (obeny–unteny) berechnet. Der errechnete Wert wird als deltay2 unter der Adresse DY2 gespeichert.

Jetzt wird HL mit unteny geladen und die Schleife in Y-Richtung – genannt QUAD_Y – betreten. In der Schleife wird zuest der Y-Wert, also das HL-Register, auf den Stapel gerettet und DE aus X_INT geladen. Mit den beiden Koordinaten wird ein MOVE ausgeführt. Darauf wird der Akku mit farbe1 geladen, der Grafikstift gesetzt und der Befehl DRAWR deltax2,0 ausgeführt. Als nächstes wird der Akku mit farbe2 geladen, erneut der Grafikstift gesetzt und – mit SCHRAE – ein DRAWR deltax,deltay ausgeführt. Der Y-Wert wird vom Stapel geholt und um 2 erhöht. DE wird mit obeny geladen, die beiden Register werden vertauscht, und das Carry-Flag wird für die folgende Subtraktion gelöscht. obeny–y wird errechnet, und die beiden Register werden wieder vertauscht. Solange bei diesem Vergleich y kleiner obeny ist, erfolgt ein Sprung zum Schleifenanfang, um die Operation für den nächsten Y-Wert durchzuführen.

Beispiele für die Befehlserweiterungen

Auch zu den eingebauten Befehlserweiterungen wollen wir Ihnen einige Beispiele vorstellen. Wegen der besseren Vergleichbarkeit haben wir uns sehr stark an die Beispiele aus dem Kapitel „Erweiterungen als BASIC-Unterprogramme" angelehnt. Da jedoch keine Unterprogramme verwendet werden, brauchen auch keine Parameter vorausbesetzt zu werden. Aus diesem Grunde haben wir die Werte – meist ohne Zuhilfenahme von Variablen – direkt bei den einzelnen Befehlen angegeben.

Da die meisten Beispiele bekannt sind, wollen wir nach der Vorstellung des Listings nur noch auf einige Besonderheiten eingehen.

```
1000 '--------------------------------
1010 '---- Vorspann und Testdaten ----
1020 '--------------------------------
1030 :
1040 INK 0,10:PAPER 0
1050 INK 1,6:PEN 1
1060 INK 2,21
1070 INK 3,4
1080 :
1090 MODE 1
1100 :
2000 '--------------------------------
2010 '---- Testdaten fuer Rechteck ---
2020 '--------------------------------
2030 :
2040 FOR i=12 TO 164 STEP 5
2050     |RECHTECK,i,i,2*i,2*i,1
2060     |RECHTECK,640-2*i,i,640-i,2*i,1
```

Programm 2.30: Beispiel für die Verwendung der neuen Grafikbefehle

```
2070      |RECHTECK,640-2*i,400-i,640-i,400-2*i,1
2080      |RECHTECK,i,400-i,2*i,400-2*i,1
2090 NEXT
2100 :
2110 GOSUB 10000
2120 :
3000 '---------------------------
3010 '--- Testdaten fuer Block ---
3020 '---------------------------
3030 :
3040 |GRMODE,1
3050 :
3060 FOR i=12 TO 164 STEP 5
3070      |BLOCK,i,i,2*i,2*i,1
3080      |BLOCK,640-2*i,i,640-i,2*i,1
3090      |BLOCK,640-2*i,400-i,640-i,400-2*i,1
3100      |BLOCK,i,400-i,2*i,400-2*i,1
3110 NEXT
3120 :
3130 GOSUB 10000
3140 :
3150 '--- Zweites Beispiel ---
3160 :
3170 FOR i=0 TO 2*PI STEP PI/8
3180      |BLOCK,320+150*SIN(i),200+100*COS(i),320+300*S
IN(i),200+150*COS(i),1
3190      |BLOCK,320-150*SIN(i),200-100*COS(i),320-300*S
IN(i),200-150*COS(i),3
3200 NEXT
3210 :
3220 |GRMODE,0
3230 :
3240 GOSUB 10000
3250 :
4000 '-----------------------------
4010 '---- Testdaten fuer Quader ---
4020 '----         (leer)        ---
4030 '-----------------------------
4040 :
4050 |QUADER,50,50,150,100,60,65,1
4060 |QUADER,250,200,200,175,280,175,1
4070 |QUADER,10,390,300,300,40,270,1
4080 :
4090 GOSUB 10000
4100 :
4110 FOR i=10 TO 350 STEP 40
4120      |QUADER,i,400-i,2*i,400-1.1*i,i+30,400-1.1*i+3
0,1
4130 NEXT
4140 :
4150 GOSUB 10000
4160 :
5000 '---------------------------
5010 '--- Testdaten fuer Quader --
5020 '---         (voll)        --
5030 '---------------------------
5040 :
```

Programm 2.30: Beispiel für die Verwendung der neuen Grafikbefehle (Forts.)

```
5050 |VOLLQUADER,100,200,350,100,50,150,1,2,3
5060 |VOLLQUADER,400,390,630,250,350,200,1,2,3
5070 :
5080 GOSUB 10000
5090 :
6000 '--------------------------
6010 '--- Testdaten fuer Kreis --
6020 '---      im Bogenmass     --
6030 '--------------------------
6040 :
6050 kanf    = 0
6060 kend    = 2*PI+2*PI/75
6070 kstep   = 2*PI/75
6080 :
6090 |BOGEN.R,320,200,100,100,@kanf,@kend,@kstep,1
6100 :
6110 GOSUB 10000
6120 :
6500 '---------------------------
6510 '--- Testdaten fuer Kreis ---
6520 '---      im Gradmass      ---
6530 '---------------------------
6540 :
6550 kend    = 361
6560 kstep   = 10
6570 :
6580 |BOGEN,320,200,100,100,0,361,10,1
6590 |BOGEN.D,320,200,100,100,@kanf,@kend,@kstep,3
6600 :
6610 GOSUB 10000
6620 :
7000 '---------------------------
7010 '- Beispiele mit Teilkreisen
7020 '---------------------------
7030 :
7040 FOR i=5 TO 50 STEP 5
7050     |BOGEN,50,50,i+20,i,90,361,5,1
7060 NEXT
7070 :
7080 FOR i=5 TO 50 STEP 5
7090     |BOGEN,145,50,i+20,i*5,90,181,5,1
7100 NEXT
7110 :
7120 FOR i=325 TO 100 STEP-25
7130     |BOGEN,145,400,i*2,i,270,361,5,1
7140 NEXT
7150 :
7160 FOR i=75 TO 300 STEP 25
7170     |BOGEN,145,0,i*3,i,0,91,5,1
7180 NEXT
7190 :
7200 GOSUB 10000
7210 :
8000 ' -----------------------------
8010 ' --- Beispiel fuer Scheiben ---
8020 ' -----------------------------
8030 :
```

Programm 2.30: Beispiel für die Verwendung der neuen Grafikbefehle (Forts.)

```
8040 |SCHEIBE,320,200,320,200,2
8050 :
8060 |GRMODE,1
8070 :
8080 |SCHEIBE,160,200,160,100,1
8090 |SCHEIBE,480,200,160,100,1
8100 |SCHEIBE,320,300,160,100,1
8110 |SCHEIBE,320,100,160,100,1
8120 |SCHEIBE,320,200,130,80,3
8130 :
9000 ' Beispiele fuer Radien und Kreise
9010 ' finden Sie in Kapitel 4 (Torten-
9020 ' und Saeulendiagramme
9030 :
9998 END
9999 :
10000 '-----------------------
10010 '--- Warteschleife auf --
10020 '--- Tastendruck und   --
10030 '--- Loeschen des      --
10040 '--- Bildschirms       --
10050 '-----------------------
10060 ' GOTO 40110
10070 :
10080 LOCATE 5,24
10090 PRINT"Bitte Taste druecken"
10100 :
10110    a$=INKEY$
10120    IF a$=""THEN 10110
10130 :
10140 CLS
10150 RETURN
10160 :
```

Programm 2.30: Beispiel für die Verwendung der neuen Grafikbefehle (Forts.)

Die erste Besonderheit befindet sich in Zeile 3040, wo der Grafikmodus auf XOR-Darstellung umgeschaltet wird. Dadurch ergeben sich völlig andere Bildschirmdarstellungen, obwohl die gleichen Blöcke verwendet werden. Der XOR-Modus wird in Zeile 3220 wieder ausgeschaltet.

Beim Vollquader werden dann statt einem Quader deren zwei dargestellt, um auch die verschiedenen Perspektiven aufzuzeigen. Als Ergänzung wurde noch eine Darstellung der Scheiben ab Zeile 8000 angefügt. Beispiele für Radien und Kreise finden Sie in Kapitel 4.

Kapitel 3
Grafik mit dem Joystick

In diesem Kapitel wollen wir die in Kapitel 2 vorgestellten Unterprogramme in ein Anwendungsbeispiel einbauen. Da sicherlich nicht jeder Leser die Maschinenbefehle aus Kapitel 2 nachvollziehen wird, haben wir das Listing mit den BASIC-Unterprogrammen versehen. Dadurch, daß das ganze Listing also in BASIC geschrieben ist, ist das Zeichnen natürlich recht langsam. Diejenigen, die die Befehlserweiterung durchgeführt haben, können natürlich die entsprechenden neuen BASIC-Befehle aufrufen.

Mit dem in diesem Kapitel dargestellten Programm wollen wir Ihnen eine Anregung geben, d. h. das Programm ist zwar an sich lauffähig, kann jedoch nach Ihren Wünschen abgeändert und natürlich ergänzt werden. So haben wir z. B. nicht alle neuen Grafik-Unterprogramme verwendet, da der Einbau zusätzlicher Routinen auch für den Anfänger sehr einfach ist, wie Sie später sehen werden.

Neben der Grafik wird noch ein zusätzliches Element verwendet: der Joystick. Sofern kein Joystick zur Verfügung steht, kann natürlich auch die Eingabe über Tasten erfolgen. Hier bietet sich die Zifferntastatur wegen der Übersichtlichkeit an. Aber auch die Cursortasten können verwendet werden, wobei jedoch keine diagonalen Bewegungen möglich sind.

Mit Rücksicht auf die Portabilität des Programms haben wir in den Beispielen nur Möglichkeiten ausgewählt, die bei allen CPC-Generationen laufen. Da es sich um einen Programmvorschlag zur Ergänzung und Änderung durch den Anwender handelt, sind insbesondere die Besitzer eines 664 oder 6128 angesprochen, das Mehr an BASIC ihres Rechners auszunutzen. Sinnvoll ist der Einbau der Befehle FILL und MASK sowie die Verwendung des Farbstift-Modus.

Beginnen wir mit der Bedienungsanleitung.

3.1 BEDIENUNGSANLEITUNG

Wenn Sie das Programm starten, ist der Bildschirm in mehrer Bereiche aufgeteilt. Ca. ¼ des Bildschirms am unteren Rand ist für das Menü vorgesehen,

was Sie nach einiger Übung natürlich auch weglassen können, um die Fläche für die Grafik zu vergrößern. Allerdings kann beim Zeichnen der Grafik auch der Menübereich verwendet werden, was aber keinen guten optischen Eindruck hinterläßt. Das Menü beinhaltet folgende Punkte:

–	– Nichts	Andere Menüpunkte rückgängig machen
z	– Zeichnen	Joystick-Bewegung wird auf dem Bildschirm gezeichnet
l	– Löschen	Löschen des Bildschirms durch erneuten Programmstart
m	– Merken	Merken der aktuellen Position des Grafik-Cursors
a	– Anzeigen	Anzeigen der aktuellen Cursor-Position
h	– Hardcopy	Ausdruck der Grafik
r	– Rechteck	Rechteck ausgeben
b	– Block	Block ausgeben
q	– Quader1	Gerüst eines Quaders ausgeben
d	– Dreieck	Dreieck ausgeben
k	– Kreis	Kreis ausgeben

Das Menü kann natürlich von Ihnen noch ergänzt werden. Sowohl an der rechten als auch an der linken Bildschirmseite ist am unteren Rand je noch ein Platz dafür vorgesehen. Wird mehr Platz gebraucht, so muß das Fenster zur Menüausgabe vergrößert werden, was auch recht einfach ist, wie wir später noch sehen werden.

Weiterhin ist mitten im Menübereich (Schrift: pastellmagenta/Hintergrund: hellrot) ein relativ kleiner gelber Bereich (Balken), der zur Anzeige der aktuellen Cursor-Position dient, wie Sie durch Drücken auf die Taste a leicht feststellen können. Nach dem Programmstart wird hier die Position 320,200 angezeigt, da der Cursor in der Mitte des Bildschirms positioniert ist.

Aber Sie haben noch eine weitere Möglichkeit zur Überprüfung, wo sich der Grafik-Cursor gerade befindet: Drücken Sie den Feuerknopf an Ihrem Joystick. Sie sehen nun in der Bildschirmmitte einen kleinen Punkt aufflackern, was die aktuelle Position des Grafik-Cursors beschreibt. Lassen Sie den Feuerknopf Ihres Joysticks wieder los, so verschwindet der Punkt wieder. Wenn Sie nun den Joystick bewegen, werden Sie in dem kleinen gelben Fenster am unteren Bildschirm sehen, daß sich auch die Zahlen ändern. Optisch können Sie immer wieder die Cursor-Position durch Drücken des Joystick-Knopfes verfolgen.

Das Anzeigen der aktuellen Cursor-Position in Zahldarstellung erfordert natürlich Rechenzeit, daher können Sie durch erneutes Drücken von a das Fenster wieder löschen. Sofern Sie nicht exakt arbeiten müssen, genügt sicherlich die blinkende Anzeige des Grafik-Cursors. Wer will, kann sich hier auch ein kleines Kreuzchen anzeigen lassen, was natürlich Rechenzeit erfordert, da dieses Kreuzchen immer wieder gesetzt und gelöscht werden muß.

Geben Sie z ein, so wird die Bahn des Cursors gelb auf blau nachgezeichnet. Hier können Sie auch den Geschwindigkeitsunterschied mit/ohne Anzeige der Zahldarstellung feststellen. Unterhalb des gelben Balkens erscheint der Schriftzug „Zeichnen", der Ihnen die aktuelle Tätigkeit mitteilt. Die Schrift ist hellrot auf pastellmagenta Hintergrund.

Bei Eingabe von l wird das Programm neu gestartet, d. h. die Grafik gelöscht, eventuelle Merker zurückgesetzt (siehe unten) und das Menü angezeigt.

Drücken Sie die Taste m, so erscheint unter dem gelben Balken für die Position des Grafik-Cursors:

gemerkt:
X : 320
Y : 200

Die Koordinaten 320,200 erscheinen natürlich nur, wenn sich der Cursor auch im Bildschirmmittelpunkt befindet. Ansonsten geben die Zahlen die Position des jeweiligen gemerkten Punktes an. Besonders für die im folgenden beschriebenen Figuren ist der Merker wichtig, da zumindest immer ein weiterer Punkt zusätzlich zur aktuellen Cursor-Position bekannt sein muß, um die Figur zeichnen zu können.

Bei Drücken von h erfolgt die Ausgabe der Grafik auf dem Drucker, wobei an dieser Stelle jedoch angemerkt sein muß, daß das am Ende beschriebene Hardcopy-Programm vorher natürlich geladen sein sollte.

Kommen wir nun zu den einzelnen vorgesehenen Figuren, die wir mit den Unterprogrammen aus Kapitel 2 erstellen wollen. Die Liste der Figuren erhebt natürlich keinen Anspruch auf Vollständigkeit und kann von Ihnen beliebig – je nach Anwendungszweck – ausgebaut werden.

Bewegen Sie den Cursor zum Beispiel zum Punkt 200,150. Drücken Sie nun die Taste m, und es erscheint – wie eben beschrieben – in dem Feld unter dem gelben Balken erneut die aktuelle Cursor-Position. Bewegen Sie jetzt den Cursor auf den Punkt 400,350.

Wenn Sie nun die Taste r drücken, erscheint ein Rechteck zwischen den Koordinaten, die Sie eben angewählt haben. Die zuerst gemerkte Position geht dabei nicht verloren, kann also weiter genutzt werden, indem Sie den Cursor an eine andere Position bewegen und erneut r drücken.

Sie können aber auch gleich anschließend b drücken, und das Rechteck wird mit Farbe gefüllt, da mit den gleichen Positionen wie beim Rechteck nun ein Block ausgegeben wird.

Wenn Sie jetzt ein q drücken, tut sich nichts weiter, als daß in dem Anzeigefeld

in der Bildschirmmitte unten die Meldung „Quader (leer)“ erscheint. Wie Sie sicher noch aus Kapitel 2 wissen, sind zum Zeichnen eines Quaders drei Koordinatenangaben erforderlich. Mit dem ersten Drücken von q haben Sie nun einen zweiten Merker gesetzt. Bewegen Sie nun den Cursor zur Koordinate (450,380), und drücken Sie erneut q. Sie sehen nun den Ausschnitt eines Quaders.

Um sich die Zuordnung der gemerkten Punkte zu den Bezugskoordinaten des Quaders zu verdeutlichen, drücken Sie nun l und nach dem Löschen des Bildschirms a. Als nächstes drücken Sie m, womit die Bildschirmkoordinaten des Mittelpunktes als erster Merkpunkt ausgesucht werden. Wenn Sie nun zu den Koordinaten (370,150) gehen und q drücken, wird der zweite Punkt gemerkt. Gehen Sie dann zu den Koordinaten (340,220), und drücken Sie erneut q. Nun ist deutlich, wie die Zuordnung Koordinaten/Bezugspunkte des Quaders aussieht: Der zuerst gemerkte Punkt ergibt die Koordinaten (entsprechend Kapitel 2) obenvx und obenvy, der zweite gemerkte Punkt (erstmaliges Drücken von q) ergibt die Koordinaten hintenx und hinteny, und beim zweiten Drükken von q werden die Koordinaten untenvx und untenvy gesetzt.

Hierzu noch ein weiteres Beispiel: Nachdem Sie die Grafik erneut gelöscht haben, steuern Sie nacheinander die Koordinaten (200,350)/(220,320) und (440,150) an, wobei Sie die Tasten m, q und q in dieser Reihenfolge drücken. Sie erhalten nun das Bild eines Quaders in der üblichen Ansicht.

Neu gegenüber den Unterprogrammen aus Kapitel 2 ist die Darstellung eines Dreiecks. Ein eigenes Unterprogramm lohnt den Aufwand nicht, da einerseits drei Hilfspunkte gegeben sein müssen und andererseits diese lediglich untereinander zu verbinden sind. Auch hier wird beim erstmaligen Drücken von d zunächst ein weiterer Hilfspunkt festgelegt und erst beim nochmaligen Drükken von d das Dreieck gezeichnet. Sicherlich könnte man auch dies mit einem eigenen Befehl „Linie“ realisieren, wobei zwischen einem gemerkten Punkt und der aktuellen Position des Grafik-Cursors eine Linie gezogen wird. Das Ziehen einer Linie haben wir wegen der einfachen Durchführung für Sie als Ergänzung vorgesehen. Durch das Merken der beiden ersten Koordinaten des Dreiecks lassen sich jedoch einige Formen von Grafiken einfach erzeugen, wie Abb. 3.1 zeigt. Beim rechten Teil der Hardcopy wurde aufgrund der Koordinatenangaben in Zahlen ein regelmäßiges Muster erzeugt. Aber auch unregelmäßige Muster sind sehr leicht zu erstellen, wie der linke Teil von Abb. 3.1 zeigt.

Als letztes haben wir für Sie das Zeichnen eines Kreises vorgesehen, da hier einige Schwierigkeiten bei der Parameterwahl zu berücksichtigen sind. Um mit möglichst wenigen Hilfspunkten auszukommen, haben wir zur Darstellung eines Kreises/einer Ellipse einen ungewöhnlichen Weg gewählt: Durch

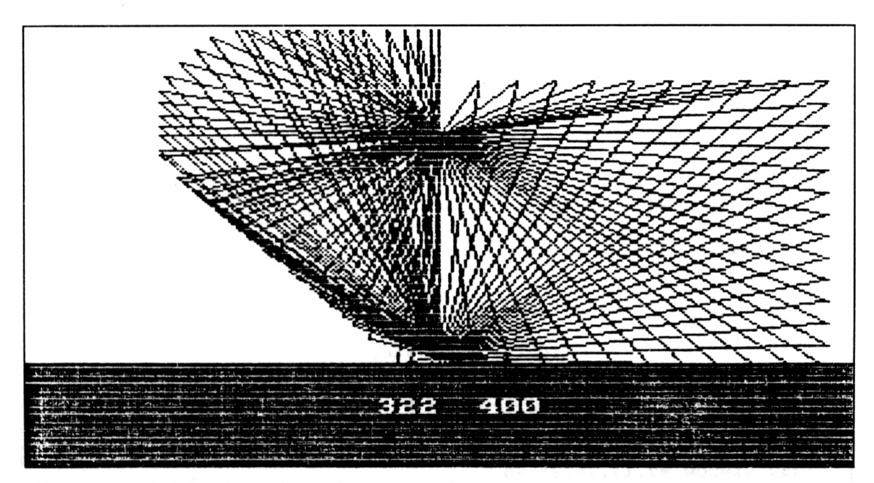


Abb. 3.1: Mit dem folgenden Programm erzeugte Grafik

den zuerst gemerkten Punkt und die aktuelle Grafik- Cursorposition ist – wie bekannt – ein Rechteck beschrieben.

In dieses Rechteck wird nun ein Kreis/eine Ellipse eingefügt, so daß sich das imaginäre Rechteck und der Kreis/die Ellipse jeweils bei 0, 90, 180 und 270 Grad berühren. Dadurch sind natürlich „schrägliegende" Ellipsen nicht möglich. Mit dem vorhandenen Wissen, besonders über die Vorgehensweise beim Merken von Hilfspunkten, dürfte es für Sie ein leichtes sein, hier eine andere Form zu wählen.

Auch die in Kapitel 2 noch vorgestellten weiteren geometrischen Figuren sollten Sie – je nach Belieben – in das Programm einbauen. Weitere Zusatzfunktionen (z. B. das Löschen von Bildschirmbereichen oder Figuren – siehe auch Kapitel 5) dürften auch nicht weiter schwierig zu realisieren sein.

3.2 LISTING

```
1000 REM ------------------------------
1010 REM ------------------------------
1020 REM ---                        ---
1030 REM ---  Zeichnen mit Joystick ---
1040 REM ---                        ---
1050 REM ------------------------------
1060 :
1070 REM --- Bildschirm einstellen  ---
```

Programm 3.1: Zeichnen mit dem Joystick

```
1080 REM --- und Fenster definieren ---
1090 :
1100 MODE 1
1110 BORDER 0
1120 INK 2,17
1130 CLS
1140 :
1150 WINDOW #1,1,40,20,25
1160 PEN #1,2
1170 PAPER #1,3
1180 CLS #1
1190 :
1200 REM ---    Menue anzeigen      ---
1210 :
1220 PRINT#1,"´-´ - nichts  h - Hardcopy  r - Recht.
1230 PRINT#1," z  - zeichen               b - Block
1240 PRINT#1," l  - loeschen              q - Quaderl
1250 PRINT#1," m  - merken                d - Dreieck
1260 PRINT#1," a  - anzeigen              k - Kreis
1270 :
1280 WINDOW #2,17,26,23,25
1290 PEN #2,3
1300 PAPER #2,2
1310 :
1320 WINDOW #3,17,26,22,22
1330 PEN #3,0
1340 PAPER #3,1
1350 CLS #3
1360 :
1370 x=320: y=200
1380 ORIGIN 0,0
1390 :

2000 REM ----------------------------
2010 REM --- Schleife zur Joystick- ---
2020 REM ---        abfrage         ---
2030 REM ----------------------------
2040 :
2050 jo=JOY(0)
2060 taste=0
2070 :
2080 IF jo>15 THEN jo=jo-16 : taste=1
2090 :
2100 IF jo=9 THEN x=x+1 : y=y+1 : GOTO 2190
2110 IF jo=10 THEN x=x+1 : y=y-1 : GOTO 2190
2120 IF jo=6 THEN x=x-1 : y=y-1 : GOTO 2190
2130 IF jo=5 THEN x=x-1 : y=y+1 : GOTO 2190
2140 IF jo=1 THEN y=y+1 : GOTO 2190
2150 IF jo=8 THEN x=x+1 : GOTO 2190
2160 IF jo=2 THEN y=y-1 : GOTO 2190
2170 IF jo=4 THEN x=x-1 : GOTO 2190
2180 :
2190 IF modus$="z" THEN PLOT x,y,1
2200 IF anzeig$="a" THEN LOCATE #3,1,1 : PRINT#3,x;y;
2210 IF taste THEN IF TEST(x,y)=0 THEN IF modus$="" OR
modus$="m" THEN PLOT X,Y,1
2220 IF taste THEN IF TEST(x,y)<>0 THEN IF modus$="" OR
 modus$="m" THEN PLOT X,Y,0
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

```
2230 :
3000 REM ------------------------------
3010 REM ---     Tastaturabfrage      ---
3020 REM ------------------------------
3030 :
3040 a$=INKEY$
3050 IF a$="" THEN 2050
3060 :
3070 IF ASC(A$)>96 OR ASC(A$)<123 THEN B$=A$ ELSE GOTO
2050
3080 :
3090 IF b$="a" AND anzeig$="" THEN anzeig$="a" : GOTO 3
110
3100 IF b$="a" AND anzeig$="a" THEN anzeig$="" : CLS #3
3110 IF B$="z" THEN GOTO 4000
3120 IF B$="l" THEN RUN
3130 IF B$="h" THEN CALL &A080 : GOTO 2050
3140 IF B$="-" THEN 4500
3150 IF B$="m" THEN 5000
3160 IF B$="r" THEN 5500
3170 IF B$="b" THEN 6000
3180 IF B$="q" THEN 6500
3190 IF B$="k" THEN 7000
3200 IF B$="d" THEN 7500
3210 :
3220 GOTO 2050
3230 :
4000 REM ------------------------------
4010 REM ---          Zeichnen        ---
4020 REM ------------------------------
4030 :
4040 CLS #2
4050 LOCATE #2,2,2
4060 :
4070 PRINT #2,"Zeichnen
4080 :
4090 PEN #1,2
4100 PAPER #1,3
4110 :
4120 modus$="z"
4130 GOTO 2050
4140 :
4500 REM ------------------------------
4510 REM ---          Nichts          ---
4520 REM ------------------------------
4530 :
4540 CLS #2
4550 LOCATE #2,2,2
4560 :
4570 PRINT #2," Nichts
4580 :
4590 PEN #1,2
4600 PAPER #1,3
4610 :
4620 modus$=""
4630 GOTO 2050
4640 :
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

```
5000 REM ------------------------------
5010 REM ---     Punkt merken       ---
5020 REM ------------------------------
5030 :
5040 CLS #2
5050 LOCATE #2,1,2
5060 :
5070 PRINT #2," gemerkt:"
5080 PRINT #2,"  X:";x
5090 PRINT #2,"  Y:";y
5100 :
5110 PEN #1,2
5120 PAPER #1,3
5130 :
5140 merkx=x
5150 merky=y
5160 :
5170 modus$="m"
5180 GOTO 2050
5190 :
5500 REM ------------------------------
5510 REM ---         Rechteck       ---
5520 REM ------------------------------
5530 :
5540 CLS #2
5550 LOCATE #2,2,2
5560 :
5570 PRINT #2," Rechteck "
5580 :
5590 PEN #1,2
5600 PAPER #1,3
5610 :
5620 obenx=merkx
5630 obeny=merky
5640 untenx=x
5650 unteny=y
5660 farbe=1
5670 :
5680 GOSUB 50000
5690 :
5700 GOTO 2050
5710 :
6000 REM ------------------------------
6010 REM ---           Block        ---
6020 REM ------------------------------
6030 :
6040 CLS #2
6050 LOCATE #2,2,2
6060 :
6070 PRINT #2,"  Block"
6080 :
6090 PEN #1,2
6100 PAPER #1,3
6110 :
6120 obenx=merkx
6130 obeny=merky
6140 untenx=x
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

```
6150 unteny=y
6160 farbe=1
6170 :
6180 GOSUB 51000
6190 :
6200 GOTO 2050
6210 :
6500 REM ------------------------------
6510 REM ---        Quader (leer)     ---
6520 REM ------------------------------
6530 :
6540 CLS #2
6550 LOCATE #2,2,2
6560 :
6570 PRINT #2,"  Quader"
6580 PRINT #2,"  (leer)"
6590 :
6600 PEN #1,2
6610 PAPER #1,3
6620 :
6630 IF modus$="m" THEN hintenx=x : hinteny=y : modus$=
"" : GOTO 2050
6640 :
6650 obenvx=merkx
6660 obenvy=merky
6670 untenvx=x
6680 untenvy=y
6690 farbe=1
6700 :
6710 GOSUB 52000
6720 :
6730 GOTO 2050
6740 :
7000 REM ------------------------------
7010 REM ---          Kreis           ---
7020 REM ------------------------------
7030 :
7040 CLS #2
7050 LOCATE #2,2,2
7060 :
7070 PRINT #2," Kreis"
7080 :
7090 PEN #1,2
7100 PAPER #1,3
7110 :
7120 mittelx=merkx
7130 mittely=merky
7140 radiusx=ABS(x-merkx)
7150 radiusy=ABS(y-merky)
7160 kend   =360
7170 kstep  =10
7180 kfarbe =1
7190 :
7200 GOSUB 55000
7210 :
7220 GOTO 2050
7230 :
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

```
7500 REM ------------------------------
7510 REM ---       Dreieck          ---
7520 REM ------------------------------
7530 :
7540 CLS #2
7550 LOCATE #2,2,2
7560 :
7570 PRINT #2,"Dreieck"
7580 :
7590 PEN #1,2
7600 PAPER #1,3
7610 :
7620 IF modus$="m" THEN merkx2=x : merky2=y : modus$=""
 : GOTO 2050
7630 :
7640 MOVE merkx,merky
7650 DRAW merkx2,merky2,1
7660 DRAW x,y,1
7670 DRAW merkx,merky,1
7680 :
7690 GOTO 2050
7700 :
50000 REM *****************************
50010 REM *****************************
50020 REM ***                       ***
50030 REM ***   Grafische           ***
50040 REM ***                       ***
50050 REM ***      Unterprogramme   ***
50060 REM ***                       ***
50070 REM *****************************
50080 REM *****************************
50090 :
50100 :
50110 :
50120 REM -----------------------------
50130 REM ---      Rechteck         ---
50140 REM -----------------------------
50150 :
50160    MOVE obenx,obeny
50170    DRAW obenx,unteny,farbe
50180    DRAW untenx,unteny,farbe
50190    DRAW untenx,obeny,farbe
50200    DRAW obenx,obeny,farbe
50210 :
50220 RETURN
50230 :
51000 REM -----------------------------
51010 REM ---       Block           ---
51020 REM -----------------------------
51030 :
51040    FOR block=obenx TO untenx
51050       MOVE block,obeny
51060       DRAW block,unteny,farbe
51070    NEXT
51080 :
51090 RETURN
51100 :
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

```
52000 REM -----------------------------
52010 REM ---       Quader leer       ---
52020 REM -----------------------------
52030 :
52040 REM ---   vorderes Rechteck    ---
52050 :
52060    obenx  = obenvx
52070    obeny  = obenvy
52080    untenx = untenvx
52090    unteny = untenvy
52100    GOSUB 50120
52110 :
52120 REM ---   hinteres Rechteck    ---
52130 :
52140    obenx  = hintenx
52150    obeny  = hinteny
52160    untenx = untenvx+hintenx-obenvx
52170    unteny = untenvy+hinteny-obenvy
52180    GOSUB 50120
52190 :
52200 REM ---   restliche Linien     ---
52210 :
52220    MOVE obenvx,obenvy
52230    DRAW hintenx,hinteny,farbe
52240 :
52250    MOVE untenvx,obenvy
52260    DRAW untenvx+hintenx-obenvx,hinteny,farbe
52270 :
52280    MOVE untenvx,untenvy
52290    DRAW untenvx+hintenx-obenvx,untenvy+hinteny-ob
envy,farbe
52300 :
52310    MOVE obenvx,untenvy
52320    DRAW hintenx,untenvy+hinteny-obenvy,farbe
52330 :
52340 RETURN
52350 :
53000 REM -----------------------------
53010 REM ---       Quader voll       ---
53020 REM -----------------------------
53030 :
53040    obenx  = obenvx
53050    obeny  = obenvy
53060    untenx = untenvx
53070    unteny = untenvy
53080    farbe=farbel
53090    GOSUB 51000
53100 :
53110    FOR quader=obenvx TO untenvx
53120       MOVE quader,obenvy
53130       DRAW quader+hintenx-obenx,hinteny,farbe2
53140    NEXT
53150 :
53160    FOR quader=untenvy TO obenvy
53170       MOVE untenvx,quader
53180       DRAW untenvx+hintenx-obenvx,quader+hinteny-
obenvy,farbe3
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

```
53190     NEXT
53200 :
53210     MOVE obenvx,obenvy
53220     DRAW untenvx,obenvy,0
53230     DRAW untenvx,untenvy,0
53240     MOVE untenvx,obenvy
53250     DRAW untenvx+hintenx-obenvx,hinteny,0
53260 :
53270 RETURN
53280 :
54000 REM -----------------------------
54010 REM ---    Kreis im Bogenmass   ---
54020 REM -----------------------------
54030 :
54040     FOR klauf=kanf TO kend STEP kstep
54050        punktx=mittelx+radiusx*SIN(klauf)
54060        punkty=mittely+radiusy*COS(klauf)
54070        IF klauf=kanf                                 T
HEN MOVE punktx,punkty :                          GOTO 54090
54080        DRAW punktx,punkty,kfarbe
54090     NEXT
54100 :
54110 RETURN
54120 :
55000 REM -----------------------------
55010 REM ---    Kreis im Gradmass    ---
55020 REM -----------------------------
55030 :
55040     DEG
55050 :
55060     GOSUB 54000
55070 :
55080     RAD
55090 :
55100 RETURN
55110 :
56000 REM -----------------------------
56010 REM ---          Radius         ---
56020 REM ---                         ---
56030 REM --- Einsprung Grad:  56070---
56040 REM --- Einsprung Bogen: 56090---
56050 REM -----------------------------
56060 :
56070     DEG
56080 :
56090     MOVE mittelx,mittely
56100 :
56110     DRAW mittelx+radiusx*SIN(winkel),mittely+radiu
sy*COS(winkel),farbe
56120 :
56130 GOTO 56160
56140     RAD
56150 :
56160 RETURN
56170 :
57000 REM -----------------------------
57010 REM ---          Stern          ---
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

```
57020 REM -----------------------------
57030 :
57040    MOVE mittelx,mittely
57050 :
57060    FOR klauf=kanf TO kend STEP kstep
57070      IF hilf=0 THEN radiusx=radiusx1 : radiusy=r
adiusy1 : hilf=1 : GOTO 57090
57080      IF hilf=1 THEN radiusx=radiusx2 : radiusy=r
adiusy2 : hilf=0
57090      punktx = mittelx+radiusx*SIN(klauf)
57100      punkty = mittely+radiusy*COS(klauf)
57110      IF erstlauf=0 THEN MOVE punktx,punkty : ers
tlauf=1 : GOTO 57130
57120      DRAW punktx,punkty,kfarbe
57130    NEXT
57140 :
57150    erstlauf = 0
57160    hilf     = 0
57170 :
57180 RETURN
57190 :
```

Programm 3.1: Zeichnen mit dem Joystick (Forts.)

3.3 PROGRAMMBESCHREIBUNG

Bevor wir zur Besprechung des Listings im einzelnen kommen, hier kurz die Aufteilung in einzelne Programmblöcke:

1000 Definieren der Fenster und Farben sowie Einstellen des Bildschirms
2000 Joystick-Abfrage
3000 Tastaturabfrage
4000 Zeichnen
4500 Nichts – in der Regel zum Löschen des Zeichen-Modus verwendet
5000 Aktuelle Cursor-Position merken
5500 Rechteck
6000 Block
6500 Quader (leer)
7000 Kreis
7500 Dreieck

50000 Grafische Unterprogramme

50120 Rechteck
51000 Block
52000 Quader (leer)
53000 Quader (voll)
54000 Kreis im Bogenmaß

55000 Kreis im Gradmaß
56000 Radius
57000 Stern

Die Grafikunterprogramme sind identisch mit den in Kapitel 2 vorgestellten Programmstücken, wobei wir die Zeilen ab 53000, 56000 und 57000 bereits für eine sinnvolle Ergänzung Ihrerseits mit übernommen haben. Außerdem haben wir sowohl das Unterprogramm zur Darstellung des Kreises im Bogenmaß als auch im Gradmaß übernommen, um Ihnen eine Auswahl nach Ihren Wünschen zu ermöglichen. Die Programmzeilen ab 50000 werden im weiteren nicht besprochen, hier sei auf Kapitel 2 nochmals hingewiesen.

Bildschirmeinstellung, Fenster definieren

Zunächst wird der Bildschirmmodus auf 40 Zeichen Bildschirmbreite und vier Farben eingestellt. Wir haben hier einen Mittelweg zwischen farblicher Darstellung und Zeichengröße gewählt. Sofern Sie mehr Farben darstellen wollen, ist einerseits in Zeile 1100 der Modus 0 einzustellen, andererseits ein Menüpunkt „Farbe ändern" zu integrieren, außerdem wird die Änderung einiger weiterer Befehle im Listing nötig, die die Farbe für das Unterprogramm angeben.

Sofern Sie auf eine farbliche Trennung zwischen Menü, Anzeigen der aktuellen Cursor-Position sowie Anzeigen der aktuellen Tätigkeit und der Grafik keinen Wert legen, können Sie auch Modus 2 wählen. Dann sind allerdings die nachfolgend beschriebenen Bildschirmfenster entsprechend anders zu dimensionieren. Der Vorteil liegt im Platzbedarf für das Menü, da natürlich in einer Zeile doppelt soviel Menüpunkte untergebracht werden können. Ein Nachteil ist allerdings die schlechtere Lesbarkeit.

Im weiteren wird lediglich die Rahmenfarbe umgestellt und die Farbauswahlnummer 2 mit der Farbe Pastelmagenta besetzt. Die anderen Farben bleiben in ihrer Grundeinstellung erhalten.

Das erste Fenster – für die Darstellung des Menüs gedacht – belegt die letzten sechs Bildschirmzeilen. Die Darstellung erfolgt mit Schrift in Pastellmagenta und hellrotem Hintergrund. In Zeile 1180 wird durch den Befehl zum Bildschirmlöschen das Fenster sofort mit hellrotem Hintergrund eingefärbt.

Zur Anzeige des Menüs ist nichts weiter zu tun, als das l im Menüpunkt Quader einzugeben. Dadurch soll angedeutet werden, daß hier ein leerer Quader (Quadergerüst) dargestellt wird. Für einen ausgefüllten Quader wäre vielleicht der Menüpunkt Q sinnvoll.

Das Fenster #2 dient zur Darstellung der aktuellen Tätigkeit und befindet sich

in der Mitte der letzten unteren drei Bildschirmzeilen. Schrift- und Hintergrundfarbe wurden gegenüber Fenster #1 vertauscht.

Das Fenster #3 belegt lediglich den Teil einer einzigen Bildschirmzeile (Zeile 22), wobei die Breite dem Bildschirmfenster #2 entspricht. Die Farbwahl erfolgt invers gegenüber der Darstellung der Grafik.

Als letzte Tätigkeit im Vorspann wird noch die aktuelle Position des Grafik-Cursors auf die Bildschirmmitte und der Koordinatenursprung auf die linke untere Bildschirmecke gelegt.

Joystick-Abfrage

Bei der Joystick-Abfrage wird zunächst der Variablen jo der aktuelle Wert des Joystick-Zustands zugewiesen, damit im folgenden keine Unstimmigkeiten auftreten können, wenn während des Programmablaufs der Wert von JOY(0) durch Bewegung des Joysticks geändert wird.

Die Variable taste ist ein Merker, der aussagt, ob die Feuertaste gedrückt ist oder nicht. Sie wird zu Anfang zurückgesetzt.

Sofern die Feuertaste gedrückt ist, wird der Merker taste auf den Wert 1 gesetzt und in der Variablen jo der Wert sechzehn subtrahiert, so daß im weiteren eine Unterscheidung der Werte jo mit/ohne Feuertaste nicht mehr nötig ist. Dies dient zur Verringerung der Rechenzeit, da gerade die Laufzeit in dem Programmteil Joystick-Abfrage und Tastaturabfrage für die Geschwindigkeit des Programms von entscheidender Bedeutung ist.

Entsprechend den Werten von jo wird die aktuelle Position des Grafik-Cursors – zunächst imaginär – geändert. Dabei werden erst die Diagonalen abgefragt, um die schnellstmögliche Bewegung zu einer angestrebten Koordinate quer über den Bildschirm zu ermöglichen. Jeder Befehl endet mit einem Sprung auf Zeile 2190, womit bei positiver Überprüfung der Rest der Zeilen bis 2180 übersprungen wird.

Ab Zeile 2190 werden bereits einige Modi überprüft. In Zeile 2190 selbst wird überprüft, ob der aktuelle Modus „Zeichnen“ ist, womit ein Punkt ausgegeben werden muß.

Soll die zahlenmäßige Anzeige der aktuellen Cursor-Position erfolgen, so wird dies in Zeile 2200 erkannt und entsprechend berücksichtigt.

Die Zeilen 2210 und 2220 beschäftigen sich mit dem Zustand der Feuertaste. Hier wurde ein kleiner Trick angewendet: Statt der Verknüpfung der verschiedenen Bedingungen mit AND wurde hier eine geschachtelte bedingte Verzweigung verwendet. Dies hat wiederum Vorteile im Rechenzeitbedarf, da

der Anweisungsteil hinter THEN nicht ausgeführt wird, wenn die Bedingung erfüllt ist. Bei einer Verwendung von AND müßte die gesamte Bedingung überprüft werden, im vorliegenden Fall wird zuerst lediglich der Zustand des Feuerknopfes berücksichtigt. Ist dieser nicht gedrückt, so wird die jeweilige Zeile schnellstmöglich verlassen.

Ist die Feuertaste gedrückt, so soll der Punkt am Bildschirm invertiert werden, was das in der Bedienungsanleitung beschriebene Blinken erzeugt. Das Blinken wird jedoch lediglich erzeugt, wenn der aktuelle Modus gleich „nichts“ oder „Koordinaten gemerkt“ ist. In Abhängigkeit davon, ob ein Bildschirmpunkt an der aktuellen Position des Grafik-Cursors gesetzt ist oder nicht, wird dieser gelöscht oder gesetzt.

Tastaturabfrage

Gegenüber der normalen Warteschleife auf einen Tastendruck wird in Zeile 3050 nicht zur Zeile 3040 gesprungen, sondern erneut zur Überprüfung des Joystick-Zustands. Dadurch ist wieder eine schnellstmögliche Verarbeitung der Eingabedaten möglich. Zeile 3070 beinhaltet eine Plausibilitätsprüfung, so daß nur die Buchstabentasten gedrückt werden können. Ist keine zulässige Taste gedrückt, so wird wieder zur Joystick-Abfrage übergegangen. Wurde ein zulässiger Wert eingegeben, so wird dieser an die Variable b$ übergeben, und in den Zeilen ab 3090 wird entsprechend verzweigt. Die Eingabe von a wird quasi als Flipflop gehandhabt, d. h. die Variable anzeig$ enthält nach dem Drücken von a ein a, wenn keins vorhanden war, und umgekehrt. Wichtig ist der Sprung zur Zeile 3110 in Zeile 3090, da sonst keine Wirkung erzielt würde. Um Irrtümer in der Bildschirmanzeige zu vermeiden, wird das Fenster #3 gelöscht, wenn eine weitere Anzeige nicht erfolgen soll.

Von den weiteren Punkten der Verzweigung sind nur die Möglichkeiten l und h interessant. Bei l wird das Programm neu gestartet, somit werden alle Variablen und der Bildschirm gelöscht. Als Erweiterung könnte man hier das Löschen von Teilbereichen des Bildschirms vorsehen. Als kleine Hilfestellung sei die Lektüre von Kapitel 5 angeraten.

Der Menüpunkt h kann natürlich nur angewählt werden, wenn das Hardcopy-Unterprogramm aus dem letzten Kapitel dieses Buchs vorher in den Speicher geladen wurde. Eventuell empfiehlt sich auch eine Verbindung des Hardcopy-Programms mit dem vorliegenden Listing.

Die einzelnen Bereiche, die durch einen Menüpunkt angewählt werden, sind nicht als Unterprogramme ausgebildet, so daß wir auch hier wieder Rechenzeit sparen, da sich der Computer keine Rücksprungadresse merken muß. In allen weiteren Programmteilen wird jeweils zum Beginn der Joystick-Abfrage gesprungen.

Zeichnen

Beim Menüpunkt Zeichnen wollen wir zunächst die allgemeine Handhabung erläutern. Beim Zeichnen und allen weiteren Programmsequenzen wird zunächst im Fenster #2 der aktuelle Arbeitszustand dargestellt. Dazu wird das Fenster #2 gelöscht und anschließend der Cursor – in der Regel – auf Position 2,2 positioniert, wonach die Ausgabe des Arbeitsvorgangs erfolgt.

Sofern Sie andere farbliche Gestaltungsmöglichkeiten gewählt haben, ist es angeraten, Stift- und Papierfarbe wieder umzuschalten. In Zeile 4120 wird der Merker modus$ noch auf z gesetzt, um dem Programmstück ab Zeile 2190 die neue Betriebsart entsprechend kundzutun.

Nichts

Dieses Programmstück ist analog den Programmzeilen ab 4000 aufgebaut. Die einzigen Änderungen befinden sich in den Zeilen 4570 (Text) und 4620 (Modus annullieren).

Punkt merken

Hier wird es etwas komplizierter als in den letzten beiden Programmstücken. Zunächst ist der Cursor auf die erste Zeile in Fenster #2 zu positionieren, da wir bei der Ausgabe etwas mehr Platz benötigen. Außerdem werden in den Zeilen 5140 und 5150 die beiden Hilfsvariablen merkx und merky mit der aktuellen Position des Grafik-Cursors besetzt, und in Zeile 5170 wird der neue Betriebszustand an die Variable modus$ übergeben.

Rechteck

In den Zeilen ab 5500 erfolgt die Vorbesetzung und der Aufruf des Unterprogramms zum Zeichnen einer Figur. Neben den üblichen Arbeiten an Fenster #2 werden ab Zeile 5620 die Eingabeparameter für das Unterprogramm gesetzt, und es wird in Zeile 5680 aufgerufen. Sofern Sie andere Farbausgaben wünschen, ist Zeile 5660 entsprechend zu ändern (Zuweisung einer Farbvariablen, die über das Menü eventuell geändert werden kann).

Block

Gegenüber dem zuletzt beschriebenen Programmstück keine wesentlichen Änderungen.

Quader (leer)

Hier haben wir das erstemal den Fall vorliegen, daß für eine geometrische Figur zwei Hilfspunkte gemerkt werden müssen. Neben dem üblichen Vorlauf

wird deshalb zunächst in Zeile 6630 geprüft, ob bereits eine Koordinate mit dem Menüpunkt m gemerkt wurde. Wenn ja, wird die aktuelle Cursor-Position den Eingabeparametern hintenx und hinteny zugewiesen und die Eingabe in der Variablen modus$ gelöscht.

Wenn nun beim nächsten Aufruf des Unterprogramms ab Zeile 6500 die Zeile 6630 erreicht wird, wird diese übergangen.

Dies ist natürlich nicht sehr anwenderfreundlich, da die Tasten in einer bestimmten Reihenfolge gedrückt werden müssen und keine Plausibilitätsprüfung erfolgt. Auf eine Plausibilitätsprüfung wurde allerdings bewußt verzichtet, da wir uns hier mit den grafischen Möglichkeiten beschäftigen. Plausibilitätsprüfungen einzubauen ist jedoch keine Schwierigkeit, wenn man z. B. die Anzahl der Aufrufe der einzelnen Tasten mitzählt.

Im weiteren werden wieder die Eingabeparameter für das Unterprogramm ab Zeile 5200 besetzt, und das Programm setzt bei der Joystick-Abfrage fort.

Kreis

Wie bereits bei der Bedienungsanleitung beschrieben, wird der Kreis als Innenkreis eines Rechtecks ausgegeben, wobei der Mittelpunkt durch die Koordinaten des zuerst gemerkten Bildschirmpunktes dargestellt wird. Die Radien ergeben sich aus dem Abstand zwischen der gemerkten und der aktuellen Position des Grafik-Cursors. Den Abstand erhält man durch die Absolut-Funktion (ABS()). Wer die Sache vereinfachen will, kann auf die Variablen kend, kanf, kstep und kfarbe verzichten. Von Hause aus wurde bereits die Variable kanf weggelassen, da sie durch den Programmstart sowieso auf 0 gesetzt ist. Bei Verzicht auf die vorgenannten Variablen sind konstante Werte in das entsprechende Unterprogramm einzutragen.

Andererseits können diese Werte auch mittels eines INPUT-Befehls vom Bildschirm abgefragt werden (am besten ein Fenster #4 am unteren Bildschirmrand definieren oder das Fenster für die Tätigkeitsanzeige verwenden), wodurch auch Halbkreise oder n-Ecke ausgegeben werden können.

Dreieck

Neben dem üblichen Ablauf zur Darstellung der aktuellen Tätigkeit in Fenster #2 wird – analog zur Vorgehensweise beim Quader – beim erstmaligen Drücken von d ein zweiter Merkpunkt gesetzt, der in den Hilfsvariablen merkx2 und merky2 abgelegt wird.

Ab Zeile 7640 wird dann das Dreieck ausgegeben.

Grafische Unterprogramme

Die grafischen Unterprogramme sollen an dieser Stelle nicht besprochen werden. Hier sei nochmals auf Kapitel 2 verwiesen.

3.4 VARIABLENÜBERSICHT

a$	Hilfsvariable zum Einlesen eines Zeichens von der Tastatur
anzeig$	Aktueller Zustand für Zahlanzeige der Koordinaten des Grafik-Cursors " " = nicht anzeigen "a" = anzeigen
b$	Wert von a$, wenn gültiges Zeichen
farbe	Eingabeparameter für UP Rechteck/Block
hintenx	Eingabeparameter für UP Quader
hinteny	Eingabeparameter für UP Quader
jo	Aktueller Zustand des Joysticks (ab Zeile 2080 gegebenenfalls vermindert um den Wert für die Feuertaste)
kend	Eingabeparameter für UP Quader
kfarbe	Eingabeparameter für UP Quader
kstep	Eingabeparameter für UP Quader
merkx	Gemerkte X-Koordinate
merky	Gemerkte Y-Koordinate
merkx2	Hilfsvariable zum Merken der zweiten X-Koordinate beim Zeichnen des Dreiecks
merky2	Hilfsvariable zum Merken der zweiten Y-Koordinate beim Zeichnen des Dreiecks
mittelx	Eingabeparameter für UP Kreis
mittely	Eingabeparameter für UP Kreis
modus$	Merker für aktuellen Arbeitszustand
obenvy	Eingabeparameter für UP Quader
obenvy	Eingabeparameter für UP Quader
obenx	Eingabeparameter für UP Rechteck/Block (oben links)
obeny	Eingabeparameter für UP Rechteck/Block (oben links)
radiusx	Eingabeparameter für UP Kreis
radiusy	Eingabeparameter für UP Kreis
taste	Aktueller Zustand der Feuertaste
untenvx	Eingabeparameter für UP Quader
untenvy	Eingabeparameter für UP Quader
untenx	Eingabeparameter für UP Rechteck/Block (unten rechts)
unteny	Eingabeparameter für UP Rechteck/Block (unten rechts)
x	X-Position des aktuellen Grafik-Cursors
y	Y-Position des aktuellen Grafik-Cursors

Kapitel 4
Diagramme

Diagramme werden hauptsächlich zur Veranschaulichung von Zahlenkolonnen z. B. bei Statistiken verwendet. Niemand bestreitet sicherlich, daß ein Diagramm zahlenmäßige Zusammenhänge im groben besser veranschaulicht als endlose Zahlenkolonnen. Gegenüber der Exaktheit der Zahlen – auf die es in den meisten Fällen dann jedoch auch nicht ankommt – haben sie zwar einen Nachteil, dies wird aber um ein Vielfaches durch die bessere Übersichtlichkeit aufgewogen.

Mit Diagrammen lassen sich einerseits zeitlich zurückliegende Zahlen als Statistiken sehr gut darstellen (Umsatzstatistiken, Kostenstatistiken, Ertragsstatistiken, Verkaufsstatistiken etc.), aber auch kombinierte Aussagen (z. B. durchschnittliches Einkommen für Arbeiter, Angestellte, Selbständige und Beamte) machen. Auch Prognosen in die Zukunft werden durch Diagramme sehr erleichtert, wenn dies auch nur in eingeschränkter Form gilt, da es das Wesen der Zukunft ist, daß man sie nicht kennt. Trotzdem lassen sich anhand von Kurvenverläufen relativ sichere Prognosen stellen.

Im folgenden wollen wir mehrere der Diagrammformen vorstellen. Zunächst die häufigste Erscheinungsform: das Liniendiagramm. Anschließend gehen wir auf die Balkendiagramme ein, wobei wir hier wiederum drei verschiedene Darstellungsarten aufzeigen. Den Abschluß bilden Hinweise zur Erstellung von Säulen- und Tortendiagrammen.

4.1 LINIENDIAGRAMME

Liniendiagramme stellen die häufigst verwendete Diagrammform dar. Sie werden zweckmäßigerweise für Daten verwendet, die einen gewissen kontinuierlichen Trend wiedergeben sollen (z. B. Umsätze). Durch die fortlaufende Linienführung lassen sich auch zurückliegende oder zukünftige Trends durch die Anschaulichkeit leicht ermitteln.

Werden in einem Liniendiagramm mehrere Linien gegenübergestellt, was besonders aus Vergleichsgründen sehr häufig vorkommt, können diese durch unterschiedliche Farben ausgewiesen werden. Besitzer eines CPC 664 oder ei-

nes CPC 6128 haben weiterhin auch die Möglichkeit, mit dem MASK-Befehl gestrichelte Linien nach eigenen Wünschen zu kreieren.

Im folgenden Programmlisting haben wir die Möglichkeit des Liniendiagramms aufgezeigt, wobei hier auch drei verschiedene Zahlengruppen als Liniendiagramm wahlweise dargestellt werden können. Dies wird durch den RESTORE-Befehl sehr einfach möglich. Als weitere Besonderheit haben wir den TAG-Befehl verwendet, mit dem es möglich ist, auch Beschriftungen genau entsprechend der Grafik zu plazieren. Weiterhin wird die Möglichkeit vorgestellt, auch ohne den TAG-Befehl eine Beschriftung durchzuführen, indem durch geschickte Wahl des Abbildungsmaßstabs in bezug auf die Größe der textlichen Darstellung eingegangen wird.

```
1000 REM ------------------------------
1010 REM --- Vorspann und Daten     ---
1020 REM ------------------------------
1030 :
1040 REM --- Texte der Ueberschrift ---
1050 :
1060 DATA 1/84,2/84,3/84,4/84,1/85,2/85
1070 :
1080 DIM text$(6)
1090 :
1100 FOR i=1 TO 6
1110    READ text$(i)
1120 NEXT
1130 :
1140 REM ---        Menue           ---
1150 :
1160 CLS
1170 :
1180 LOCATE 5,5
1190 PRINT "1 - UMSATZ IN STUECK"
1200 :
1210 LOCATE 5,7
1220 PRINT "2 - UMSATZ IN DM"
1230 :
1240 LOCATE 5,9
1250 PRINT "3 - ROHERTRAG"
1260 :
1270 a$=INKEY$
1280 IF a$="" THEN GOTO 1270
1290 IF A$="1" THEN RESTORE 2060
1300 IF a$="2" THEN RESTORE 2320
1310 IF a$="3" THEN RESTORE 2620
1320 :
2000 REM ------------------------------
2010 REM ---        Daten          ---
2020 REM ------------------------------
```

Programm 4.1: Liniendiagramme

```
2030 :
2040 REM ---    Stueckumsatz          ---
2050 :
2060 DATA 5,6
2070 :
2080 DATA 41.6666667,90
2090 :
2100 DATA 800,2300,3500,4800,5200,5300
2110 DATA 0,900,2000,3500,4000,4200
2120 DATA 0,100,3000,6000,9000,12000
2130 DATA 0,0,0,500,1000,1400
2140 DATA 0,0,900,3400,2800,4500
2150 :
2160 DATA 14,24
2170 :
2300 REM ---    DM-Umsatz             ---
2310 :
2320 DATA 5,6
2330 :
2340 DATA 1250,90
2350 :
2360 DATA 17000,50000,90000,120000,130000,140000
2370 DATA 0,26000,70000,105000,120000,130000
2380 DATA 0,3000,135000,235000,300000,380000
2390 DATA 0,0,40000,100000,135000,170000
2400 DATA 0,0,20000,85000,60000,95000
2410 :
2420 DATA 22,16
2430 :
2600 REM ---       Rohertrag          ---
2610 :
2620 DATA 5,6
2630 :
2640 DATA 156.25,90
2650 :
2660 DATA 2000,8000,9000,12000,12500,13400
2670 DATA 0,2000,12000,12500,13000,14000
2680 DATA 0,0,5000,10000,20000,45000
2690 DATA 0,0,2000,11000,14000,15000
2700 DATA 0,0,2000,10000,12000,15000
2710 :
2720 DATA 10,32
2730 :
3000 REM -----------------------------
3010 REM --- Einlesen der Daten    ---
3020 REM -----------------------------
3030 :
3040 READ n,m
3050 READ vert,hor
3060 :
3070 DIM werte(n,m)
3080 :
3090 FOR i=1 TO n
3100    FOR j=1 TO m
3110       READ werte(i,j)
3120       werte(i,j)=werte(i,j)/vert
```

Programm 4.1: Liniendiagramme (Forts.)

```
3130    NEXT
3140 NEXT
3150 :
4000 REM ------------------------------
4010 REM ---      Hauptprogramm     ---
4020 REM ------------------------------
4030 :
4040 CLS
4050 ORIGIN 0,0
4060 farbe=1
4070 :
4080 REM ---   Kurven zeichnen      ---
4090 :
4100 FOR i=1 TO n
4110    FOR j=1 TO m
4120       DRAW posx+hor,werte(i,j),farbe
4130       posx=XPOS
4140    NEXT
4150    PLOT 0,0
4160    posx=0
4170    IF i=2 THEN INK 2,26 : farbe=2
4180    IF i=4 THEN INK 3,15 : farbe=3
4190 NEXT
4200 :
4210 PEN 4
4220 :
4230 REM ---    Vertikale Linien    ---
4240 :
4250 FOR i=0 TO m
4260    PLOT i*hor,0
4270    DRAW i*hor,350
4280    PLOT 0,0
4290    DRAW m*hor,0
4300 NEXT
4310 :
4320 REM ---   Horizontale Linien   ---
4330 :
4340 READ abschnitte,hoehe
4350 :
4360 FOR i=1 TO abschnitte
4370    PLOT 0,i*hoehe
4380    DRAW 540,i*hoehe
4390 NEXT
4400 :
4410 REM ---      Beschriftung      ---
4420 :
4430 TAG
4440 PEN 1
4450 :
4460 versatz=INT((hor-64)/2)+1
4470 FOR i=0 TO m-1
4480    MOVE i*hor+versatz,370
4490    PRINT text$(i+1);:REM';'wichtig
4500 NEXT
4510 :
4520 TAGOFF
4530 :
```

Programm 4.1: Liniendiagramme (Forts.)

```
4540 a=VAL(a$)
4550 ON a GOTO 5120,5020,5220
4560 :
5000 REM -   Beschr.  DM-Umsatz      ---
5010 :
5020 FOR i=0 TO 21
5030    LOCATE 36,25-i
5040    PRINT USING "###" ;i*20;
5050    PRINT "T"
5060 NEXT
5070 :
5080 GOTO 6020
5090 :
5100 REM -   Beschr. Stueck-Umsatz   ---
5110 :
5120 FOR i=0 TO 7
5130    LOCATE 36,25-i*3
5140    PRINT USING "##" ;i*2;
5150    PRINT " T"
5160 NEXT
5170 :
5180 GOTO 6020
5190 :
5200 REM -   Beschr. Rohertrag       ---
5210 :
5220 FOR i=0 TO 10
5230    LOCATE 36,25-i*2
5240    PRINT USING "###" ;i*5;
5250    PRINT "T"
5260 NEXT
5270 :
6000 REM ---     Warteschleife       ---
6010 :
6020 a$=INKEY$ : IF a$="" THEN 6020
6030 MODE 1
6040 INK 1,24 : PEN 1
6050 RUN
```

Programm 4.1: Liniendiagramme (Forts.)

Da wir Umsätze bzw. Ertrag als Liniendiagramm darstellen wollen, ist es zweckmäßig, als Spaltenüberschrift jeweils die Bezeichnung einer Periode zu wählen. Weil die Perioden für alle drei Liniendiagramme (Umsatz in Stück, Umsatz in DM, Rohertrag) gleich sind, muß das Erfassen dieser Überschriften – sofern sie nicht im Programm direkt verankert sind – vor der Verteilung auf die einzelnen Datengruppen erfolgen.

Erfassung der Überschriften

Als Überschriften haben wir Quartale für die Perioden ausgewählt, und zwar beginnend bei Quartal 1/84 bis hin zu Quartal 2/85. Die Überschriften werden dann in die Variablen text$() übernommen.

Menü

Es folgt das Auswahlmenü für die Daten, die angezeigt werden sollen. Hierzu wird zunächst das Menü am Bildschirm dargestellt und anschließend die Tastatur abgefragt. Nach Drücken einer der Tasten 1, 2 oder 3 wird in diesem Falle kein Sprung an eine bestimmte Programmstelle durchgeführt, sondern lediglich der Zeiger zum Einlesen von Daten aus DATA-Zeilen auf eine entsprechende Zeilennummer gesetzt.

Beschreibung der Daten

Da wir den Ausdruck des Liniendiagramms so universell wie möglich halten wollen, werden nicht nur die auszugebenden Daten eingelesen, sondern auch noch andere Informationen, wie sie zur Darstellung des Hintergrundrasters nötig sind. Durch das Hintergrundraster wird eine grobe Umrechnung der durch die Linien angegebenen Daten in Zahlen wieder möglich.

Da das Programm alle drei Gruppen von Daten gleich behandeln soll, muß auch ihr Aufbau identisch sein. Die einzelnen Werte haben folgende Bedeutung:

- Anzahl der Objekte
- Anzahl der Elemente je Objekt

- Umrechnungsfaktor für die Höhe
- Abstand der Linienpunkte voneinander

- Auszugebende Daten zeilenweise je Objekt

- Anzahl der Abschnitte
- Höhe der Abschnitte in Bildschirmpunkten

Auf die Bedeutung der einzelnen Werte werden wir im weiteren Programmverlauf noch eingehen. Wie aus den Zeilen ab 2060 ersichtlich, haben wir zur Darstellung fünf Objekte gewählt, die – entsprechend der Textüberschrift – jeweils sechs Einzeldaten ausweisen.

Da die Daten in ihrer ursprünglichen Form – so wie sie aus anderen Bereichen vorliegen – eingegeben werden sollen, dadurch aber viel zu hohe Werte gegenüber der Bildschirmdarstellung aufweisen, müssen sie noch durch einen Umrechnungsfaktor dividiert werden. Ein Bildschirmpunkt entspricht also nicht einer Zahleinheit, sondern einem Bruchteil davon. Weil für alle Werte der gleiche Umrechnungsfaktor gilt, liegt somit eine maßstäbliche Darstellung vor, die an Aussagekraft nichts verliert.

Im Falle des Stückumsatzes (Zeile 2080) beträgt der Umrechnungsfaktor

$41^2/_3$. Der nächste Wert gibt die Bildschirmbreite (Anzahl der Bildpunkte) an, die für die Darstellung einer Periode vorgesehen ist.

Der Übersichtlichkeit halber wurde für jedes Objekt eine eigene DATA-Zeile vorgesehen, so daß bei der folgenden Periode möglichst wenig Änderungen entstehen. Neue Zahlen sind jeweils ans Zeilenende anzufügen.

Etwas Rechnerei ist den beiden Zahlen in Zeile 2160 zugrunde gelegt, da durch sie die waagerechten Hintergrundlinien festgelegt werden, die später die Skala zum Ablesen der Zahlenwerte bilden. Aufgrund des Umrechnungsfaktors ist es möglich, die Einteilung so zu wählen, daß die Beschriftung ohne Verwendung des TAG-Befehls erfolgen kann.

Die Datengruppen ab Zeile 2300 und ab Zeile 2600 sind analog aufgebaut.

Einlesen der Daten

Ab Zeile 3000 erfolgt das Einlesen der zuvor definierten Daten. Die Anzahl der Objekte wird in der Variablen n abgelegt und die Anzahl der Werte je Objekt in der Variablen m. Weiterhin wird der Umrechnungsfaktor in vert abgelegt und die Bildschirmbreite je Periode in hor. Das Feld werte(,) nimmt die eigentlichen Daten auf, die in den zwei geschachtelten Programmschleifen ab Zeile 3090 eingelesen und sofort umgerechnet werden (Zeile 3120).

Linien zeichnen

Nach dem Löschen des Bildschirms, dem Setzen des Koordinatenursprungs auf die linke untere Bildschirmecke und dem Vorbesetzen der Zeichenfarbe werden sogleich die einzelnen Linien gezeichnet. Dabei wird vom errechneten Wert der einen Periode eine Verbindungslinie zum errechneten Wert der nächsten Periode gezogen. Diese Arbeit wird durch eine Besonderheit sehr einfach gemacht. Nach dem Ziehen einer Linie befindet sich der Grafik-Cursor an der zuletzt gezeichneten Position, so daß lediglich die Koordinaten des nächsten Punktes angegeben werden müssen. Dies ist auch der Grund, warum je Objekt eine Linie von der ersten bis zur letzten Periode durchgezeichnet wird und nicht die Linienführung je Periode behandelt wird. Daher läuft auch die äußere Programmschleife (Beginn in Zeile 4100) bis zur Anzahl der Objekte.

Besonderes Augenmerk ist hier auf die Variable posx zu legen. Mit ihrer Hilfe wird die jeweilige X-Koordinate errechnet, die ja je Periode um einen gewissen Wert (hor) fortschreitet. In Zeile 4130 wird der Befehl XPOS zu Hilfe genommen, wodurch nach dem Zeichnen einer Linie unsere Variable posx mit dem Wert der aktuellen X-Koordinate besetzt wird, um beim nächstenmal wieder um den Wert hor vermehrt zu werden.

Selbstverständlich ist nach dem Fertigstellen einer Linie für ein Objekt der Grafik-Cursor in die Ausgangsposition zu setzen, und auch unsere Variable posx ist wieder zu annullieren.

Wenn die Objekte unterschiedlichen Bereichen angehören, ist eine unterschiedliche Farbgebung sinnvoll. Diesem Umstand wurde in den Zeilen 4170 und 4180 Rechnung getragen. Beachten Sie bitte, daß bei den Objekten 3 und 4 die Farbauswahlnummer 2 (leuchtendweiß) als Zeichenfarbe dient und bei den Objekten 5 und 6 die Farbauswahlnummer 3 (orange). Dies ist der Fall, weil die Umschaltung am Ende der äußeren FOR...NEXT-Schleife erfolgt.

Beschriftung oben und Ausgabe der Skala

Zum Beschriften der Skala wird auf die vierte Farbe umgeschaltet. Zunächst werden in der Schleife ab Zeile 4250 die senkrechten Linien gezeichnet. Hier ist also die Anzahl der Werte je Objekt maßgebend, wobei allerdings eine Linie mehr gezeichnet werden muß, als Werte vorhanden sind, um eine beidseitige Begrenzung aller Perioden zu erreichen. Wir beginnen also am linken Bildschirmrand und setzen zunächst den Grafik-Cursor an den unteren Bildschirmrand bei entsprechender horizontaler Position. Die Linie wird nicht ganz bis zum oberen Rand durchgezogen, da auch noch eine Beschriftung erfolgt.

Als nächstes erfolgt die Ausgabe der waagerechten Linien, wozu zuerst noch die Anzahl der Abschnitte und die jeweilige Höhe in Bildschirmpunkten eingelesen wird. Dann wird in der Programmschleife ab Zeile 4360 analog zur Ausgabe der senkrechten Linien verfahren.

Bei der Beschriftung wollen wir zunächst den Eintrag der Perioden vornehmen. Dazu schalten wir den CPC 464 mit dem TAG-Befehl so um, daß die Ausgabe der Schrift anhand der Position des Grafik-Cursors erfolgt. Die Position des Grafik-Cursors gibt die linke obere Ecke des Schriftzuges an.

Da wir die Beschriftung nicht an den Anfang der Periodentrennlinie setzen wollen, sondern mitten über den entsprechenden Bildschirmbereich, ist zunächst ein Wert zu errechnen, der die Mitteilung durchführt und der in der Variablen versatz abgelegt wird. Da wir einheitlich für alle Perioden vier Zeichen Überschrift haben und im Bildschirmmodus 1 arbeiten, beträgt die Breite der Schrift 64 Bildschirmpunkte.

Dies ziehen wir von der gesamten Bildschirmbreite einer Periode ab und dividieren es durch 2, wodurch wir bei einem Versatz um diesen Wert jeweils rechts und links von der Schrift zu der Begrenzung der Periode den gleichen Abstand erhalten.

In der Programmschleife ab Zeile 4470 erfolgt die Ausgabe der Schrift. Hier-

bei sind jedoch noch einige Umstände zu berücksichtigen. Da wir jeweils von der linken Begrenzung einer Periode ausgehen, muß die Programmschleife bei 0 beginnen. Die Position errechnet sich analog der Position für die senkrechten Linien plus unserem errechneten Wert versatz. Die Y-Koordinate korrespondiert zu dem Wert in Zeile 4270.

Da die Programmschleife bei 0 beginnt, muß das Argument bei text$() um 1 erhöht werden. Besonders wichtig ist das Semikolon nach der PRINT-Ausgabe. Würde dies nicht dort stehen, so brächte der Rechner noch zwei Sonderzeichen (die Werte für CHR$(13) und CHR$(10) für Carriage Return und Zeilenvorschub).

Da wir die weitere Ausgabe nicht aufgrund der Grafikposition tätigen wollen, muß der TAG-Modus abschließend wieder ausgeschaltet werden.

Beschriftung der Skala

In den Zeilen ab 5000 befindet sich die Beschriftung für den rechten Rand, wobei aufgrund des im Menü angewählten Wertes noch zu den entsprechenden Programmzeilen verzweigt wird.

Aufgrund geschickter Wahl der waagerechten Linien können wir jeweils in einer Programmschleife mittels des LOCATE-Befehls die Position des Text-Cursors anwählen. Im Fall der Beschriftung zum DM-Umsatz wird die Beschriftung von unten (Textspalte 36) nach oben jeweils ohne Leerzeile ausgeführt. Durch die Verwendung PRINT USING erhalten wir eine exakte Anordnung der auszugebenden Skalenwerte. Wegen des Umrechnungsfaktors und der Höhe der einzelnen Abschnitte sind für jede waagerechte Linie 20 000 DM Umsatz anzusetzen. Da dieser Wert bei den gewählten Größenverhältnissen nicht mehr auf den Bildschirm paßt, benutzen wir die abkürzende Schreibweise, z. B. 20T.

Nach Abschluß der Bildschirmdarstellung wird zu einer Warteschleife gesprungen, die das Programm nach Tastendruck neu startet.

Beim Stückumsatz erfolgt die Beschriftung analog zum DM-Umsatz, jedoch werden hier zwischen zwei Skalenwerten zwei Leerzeilen gelassen, die der Faktor „*3“ in Zeile 5130 angibt. Auch die Beschriftung der Skala wurde leicht verändert.

Als letztes erfolgt die Beschriftung des Liniendiagramms zum Rohertrag, wobei hier auch wieder andere Werte für den Abstand der Darstellung und Deskalierung vorliegen.

Als Abschluß noch eine Hardcopy (Abb. 4.1), die durch das beschriebene Programm erzeugt wurde.

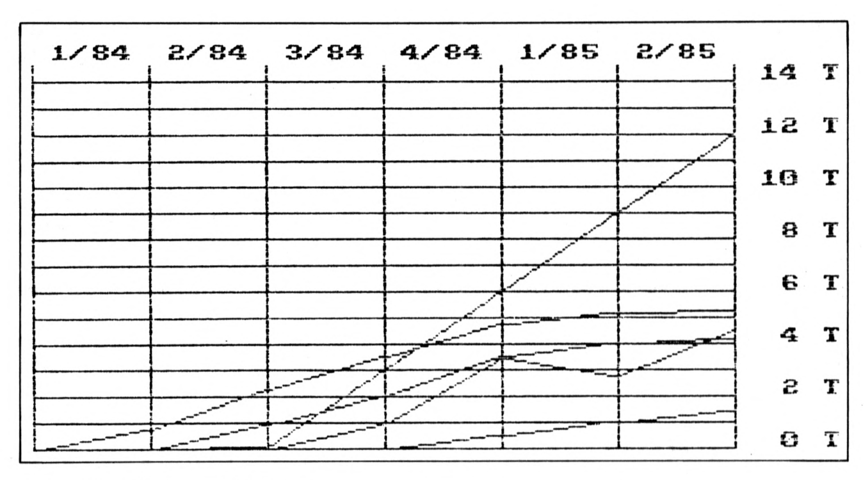


Abb. 4.1: Ein Beispieldiagramm

Hier noch eine kurze Aufstellung der verwendeten Variablen:

a	Zahlwert von A$
a$	Von Tastatur eingelesenes Zeichen
abschnitte	Anzahl der vertikalen Abschnitte
farbe	Aktuelle Farbauswahlnummer für Linien
hoehe	Höhe der einzelnen Abschnitte in Bildschirmpunkten
hor	Abstand (horizontal) der Werte
i	Laufvariable
j	Laufvariable
n	Anzahl der Objekte
m	Anzahl der Werte je Objekt
posx	Alte X-Position des Grafik-Cursors
vert	Umrechnungsfaktor für vertikale Skalierung
werte(,)	Auszugebende Werte im Original (ohne Faktor)

4.2 BALKENDIAGRAMME

Ähnlich wie bei den Liniendiagrammen kann auch die Erstellung von Balkendiagrammen erfolgen. Der Abstand der Linie von einer Grundachse wird hierbei durch die Höhe eines Balkens abgebildet. Im Gegensatz zu den Liniendiagrammen ist natürlich bei den Balkendiagrammen kein durchgehender Kurvenverlauf möglich, so daß Balkendiagramme sich mehr für diskrete Zahlengrundlagen eignen.

Balkendiagramme gibt es in verschiedenen Formen, von denen wir die drei wichtigsten vorstellen wollen. Zunächst als einfache Balken, wobei wir relativ viele Parameter zur Beeinflussung der Darstellung vorsehen werden. Im weiteren stellen wir dann diese Balken mit mehreren Farbstufen dar, wodurch noch eine Unterteilung der Summenwerte eines Balkens gegeben ist. Den Abschluß bildet eine Darstellungsform für dreidimensionale Balkendiagramme.

Die Programme zur Darstellung von Balken sind so konzipiert, daß sie auf allen Rechnern der CPC-Serie laufen. Da der CPC 464 keinen FILL-Befehl kennt, werden deshalb die Balken aus einzelnen Linien zusammengesetzt. Besitzer eines CPC 664 oder CPC 6128 können die Balken konstruieren, indem sie ein Rechteck mit der gleichen Farbe ausfüllen.

Zweidimensionale Balkendiagramme

Gehen wir zunächst auf zweidimensionale Balkendiagramme ein, die ein direktes Analogon zu den dargestellten Liniendiagrammen bilden. Hier sollen auch zunächst die Balkendiagramme ohne Farbabstufung beschrieben werden.

Zweidimensionale Balkendiagramme ohne Farbstaffelung

Bei diesem Beispiel wollen wir wegen der einfachen Struktur der Balken verschiedene Parameter aufzeigen, mit denen die Balkendiagramme anschaulich gestaltet werden können. Gegenüber den Liniendiagrammen haben wir hier die Vorgehensweise des Erfassens der Werte vom Bildschirm gewählt, wie folgendes Listing zeigt:

```
1000 REM ------------------------------
1010 REM ---    Vorspann und Daten  ---
1020 REM ------------------------------
1030 :
1040 BORDER 0
1050 CLS
1060 ORIGIN 0,0
1070 LOCATE 5,5
1080 INPUT"Anzahl der Balken";anzahl
1090 :
1100 DIM wert(anzahl)
1110 DIM hoehe(anzahl)
1120 DIM prozent(anzahl)
1130 :
1140 CLS
1150 :
1160 REM ---    Erfassen der Daten  ---
1170 :
```

Programm 4.2: Balkendiagramme

```
1180 FOR i=1 TO anzahl
1190    LOCATE 5,2+i
1200    PRINT "Wert fuer Balken";i;
1210    INPUT wert(i)
1220    summe=summe+wert(i)
1230    IF wert(i) > maximum
          THEN maximum = wert(i)
1240 NEXT
1250 :
1260 CLS
1270 LOCATE 5,2
1280 PRINT"Summe                   ";summe
1290 :
1300 LOCATE 5,4
1310 INPUT"Abstand der Balken";abstand
1320 LOCATE 5,6
1330 INPUT"Breite                  ";breite
1340 LOCATE 5,8
1350 INPUT"Grundzeile              ";grund
1360 LOCATE 5,10
1370 INPUT"Obergrenze              ";oben
1380 LOCATE 5,12
1390 INPUT"Stufenhoehe             ";stufen
1400 LOCATE 5,14
1410 INPUT"Beginn in Spalte  ";spalte
1420 LOCATE 1,16
1430 :
1440 PRINT"      1 - Rechteck
1450 PRINT"      2 - Block
1460 PRINT"      3 - Quader
1470 PRINT
1480 INPUT"      1/2/3          ";art
1490 :
1500 IF art < 1 OR art > 3
       THEN GOTO 1480
1510 IF art = 3
       THEN LOCATE 5,20 :
       INPUT"Tiefe                  ";tiefe
1520 :
1530 REM ---        Umrechnung        ---
1540 :
1550 FOR i=1 TO anzahl
1560    prozent(i) =
          INT(wert(i)*100/summe+0.5)
1570    hoehe(i) = INT(wert(i) *
          (grund+oben+tiefe+stufen *
          (anzahl-1))/maximum+0.5)
1580 NEXT
1590 :
1600 CLS
1610 :
2000 REM ------------------------------
2010 REM ---        Hauptprogramm    ---
2020 REM ------------------------------
2030 :
2040 untenx = spalte - abstand
2050 :
```

Programm 4.2: Balkendiagramme (Forts.)

```
2060 FOR i=1 TO anzahl
2070    obenx  = untenx+abstand
2080    untenx = obenx+breite
2090    obeny  = grund+hoehe(i)
2100    unteny = grund+stufen*(i-1)
2110    IF art = 3
          THEN hintenx = obenx+tiefe :
               hinteny = obeny+tiefe
2120    ON art GOSUB 3000,4000,5000
2130 NEXT
2140 :
2150 a$=INKEY$
2160 IF a$="" THEN 2150
2165 IF a$="h" THEN CALL &A080
2170 END
3000 REM ------------------------------
3010 REM ---        Rechteck         ---
3020 REM ------------------------------
3030 :
3040    MOVE obenx,obeny
3050    DRAW obenx,unteny,1
3060    DRAW untenx,unteny,1
3070    DRAW untenx,obeny,1
3080    DRAW obenx,obeny,1
3090 RETURN
3100 :
4000 REM ------------------------------
4010 REM ---          Block          ---
4020 REM ------------------------------
4030 :
4040    FOR linie=unteny TO obeny
4050       MOVE obenx,linie
4060       DRAWR breite,0,1
4070    NEXT
4080 RETURN
4090 :
5000 REM ------------------------------
5010 REM ---         QUADER          ---
5020 REM ------------------------------
5030 :
5040    MOVE obenx,obeny
5050    GOSUB 3000
5060    DRAW hintenx,hinteny,1
5070    MOVE untenx,obeny
5080    DRAW untenx+tiefe,hinteny,1
5090    MOVE obenx,unteny
5100    DRAW hintenx,unteny+tiefe,1
5110    MOVE untenx,unteny
5120    DRAW untenx+tiefe,unteny+tiefe,1
5130    obenx  = hintenx
5140    obeny  = hinteny
5150    untenx = untenx+tiefe
5160    unteny = unteny+tiefe
5170    GOSUB 3000
5180 RETURN
```

Programm 4.2: Balkendiagramme (Forts.)

Nachdem zunächst einige Vorarbeiten erledigt werden, wird die Anzahl der Balken erfaßt, und aufgrund dieser Anzahl werden die Felder wert(), hoehe() und prozent() dimensioniert. Dann wird wiederum der Bildschirm gelöscht und die Erfassung der Werte für jeden einzelnen Balken vorgenommen.

Innerhalb der Programmschleife wird zunächst der Cursor entsprechend der Laufvariablen positioniert und ein Anwenderhinweis zum aktuell zu erfassenden Balken ausgegeben. Sodann wird der Wert eingelesen und gleich der Variablen summe zugerechnet, und es wird noch getestet, ob der neue Wert größer als alle bisherigen ist. Die Variablen summe und maximum benötigen wir später, um den Maßstab ermitteln zu können.

Als Abschluß der Eingabe der auszugebenden Werte wird die Summe ausgegeben, damit man einen Anhalt für die Bildschirmdarstellung hat. Ab Zeile 1300 folgt die Erfassung der Werte zur Bildschirmdarstellung. Der Anwender kann somit den Abstand der Balken, die Breite der Balken, die Grundzeile (unterste Linie der Darstellung), die Obergrenze, die Stufenhöhe (Versatz folgender Balken nach oben) und den linken Rand (Spalte) selbst festlegen.

Außerdem kann noch die Darstellungsart als Rechteck, Block oder Quader gewählt werden, wobei bei der Darstellungsart Quader zusätzlich noch die Tiefe des Quaders erfaßt werden muß.

Ab Zeile 1550 erfolgt die Umrechnung der eingegebenen Werte, wobei in Zeile 1560 die Prozente im Gesamtanteil jedes einzelnen Balkens errechnet werden. Im vorliegenden Fall werden diese zwar nicht ausgegeben, jedoch ist die Ausgabe sinnvoll, wenn durch das Balkendiagramm die Aufteilung in verschiedene Bereiche am Gesamtanteil dargestellt wird. Besonders wichtig ist die Errechnung der Höhe, die sich aufgrund des Wertes eines Balkens und den vom Anwender erfaßten Daten ergibt. Die Höhe gibt danach den tatsächlich obersten Punkt eines Balkens aus, wobei zum Wert des Balkens natürlich der Abstand vom unteren Bildschirmrand, der Abstand vom oberen Bildschirmrand, bei einer Quaderdarstellung die Tiefe, und die Anzahl der Stufen in der Erhöhung eingehen.

Außerdem erfolgt die Umrechnung so, daß der größte Balken mit seiner Obergrenze auch die vorgegebene Obergrenze erreicht.

Im Hauptprogramm wird zunächst die linke X-Koordinate des ersten Balkens vorgegeben, wodurch in der folgenden FOR...NEXT-Schleife sehr einfach für die weiteren Balken diese Ecke errechnet werden kann (linke Ecke durch rechte Ecke ersetzen). In der Programmschleife wird dann die linke Begrenzung aus der rechten Begrenzung errechnet. Anschließend ergibt sich die rechte Begrenzung aus den Werten für die linke Begrenzung und der vorgegebenen Breite des Balkens. Der obere Wert des Balkens wird nun noch um den

Wert der Grundzeile erhöht und der untere Wert um die Stufen. Bei der Darstellung eines Quaders wird auch noch der Wert der hinteren Koordinate errechnet. Durch die Verwendung der Variablen obenx, untenx, obeny, unteny, hintenx und hinteny können wir die in Kapitel 2 beschriebenen Unterprogramme heranziehen, die ab den Zeilen 3000 angeführt sind.

Den Abschluß des Programms bildet noch eine Warteschleife, die das „ready“ am Bildschirm nach Programmabschluß verhindert. Wird während der Warteschleife ein h eingegeben, so sollte auf jeden Fall das Hardcopy-Programm geladen sein. Dieses wird dann aufgerufen.

Hier noch ein Grafik-Hardcopy-Beispiel (Abb. 4.2) und die Übersicht der Variablen.

abstand	Abstand der Balken in Bildschirmpunkten
anzahl	Anzahl der Werte (Balken)
art	Art der Ausgabe: 1 – Rechteck 2 – Block 3 – Quader
breite	Breite der Balken in Bildschirmpunkten
grund	Unterste Zeile der Ausgabe
hintenx	X-Koordinate oben links hinten für Quader
hinteny	Y-Koordinate oben links hinten für Quader
hoehe()	Höhe der Balken; aus WERT() umgerechnet
I	Laufvariable

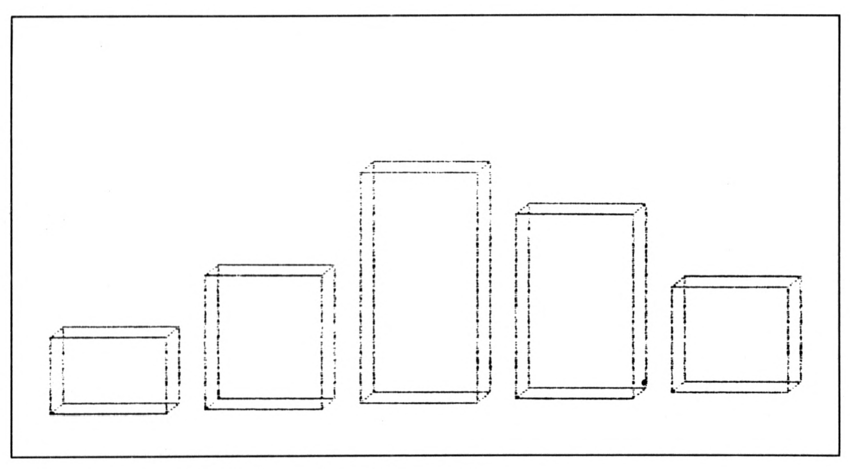


Abb. 4.2: Ein Beispiel für räumliche Balkengrafik

linie	Laufvariable bei Blockausgabe
maximum	Größter Wert
prozent()	Prozentanteil jedes einzelnen Wertes an der Summe zur weiteren Verarbeitung
oben	Oberste Zeile der Ausgabe
obenx	X-Koordinate oben links für Block, Rechteck und Quader
obeny	Y-Koordinate oben links für Block, Rechteck und Quader
spalte	Linke Grenze der Ausgabe
stufen	Erhöhung der Grundzeile für jeden weiteren Balken
summe	Aktuelle Summe/Gesamtsumme der Werte
tiefe	„seitliche Versetzung" beim Quader
untenx	X-Koordinate unten rechts für Block, Rechteck und Quader
unteny	Y-Koordinate unten rechts für Block, Rechteck und Quader
wert()	Auszugebende Werte je Balken

Zweidimensionale Balkendiagramme mit Farbstaffelung

Im folgenden Programm wollen wir auf die frei wählbare Darstellung verzichten und nur Balken mit drei Farbabstufungen darstellen, wobei wir hier wiederum auch eine Beschriftung unter Zuhilfenahme des TAG-Befehls ausführen wollen.

Abgestufte Balkendiagramme werden dort gebraucht, wo einerseits ein Gesamtüberblick mit Summenwerten gegeben sein soll, andererseits aber auch eine Aufteilung dieser Summe in grobe Bereiche.

```
1000 REM ------------------------------
1010 REM ---    Vorspann und Daten  ---
1020 REM ------------------------------
1030 :
1040 BORDER 0
1050 MODE 1
1060 :
1070 anzahl = 5
1080 :
1090 DIM wert    (anzahl,3)
1100 DIM text$   (anzahl)
1110 DIM summe   (anzahl)
1120 DIM hoehe   (anzahl,3)
1130 DIM prozent(anzahl,3)
1140 :
1150 REM ---    Einlesen der Daten   ---
1160 :
1170 DATA 1981,10000,50000,70000
1180 DATA 1982,15000,60000,80000
```

Programm 4.3: Zweidimensionale Balkendiagramme mit Farbstaffelung

```
1190 DATA 1983,17500,62500,77500
1200 DATA 1984,18000,61700,81000
1210 DATA 1985,21000,75000,90000
1220 :
1230 FOR i=1 TO anzahl
1240    READ text$(i)
1250 :
1260    FOR j=1 TO 3
1270       READ wert(i,j)
1280       hoehe(i,j)=wert(i,j)/500
1290       summe(i)=summe(i)+wert(i,j)
1300    NEXT
1310 :
1320    FOR j=1 TO 3
1330       prozent(i,j)=wert(i,j)*100/summe(i)
1340    NEXT
1350 :
1360 NEXT
1370 :
2000 REM ------------------------------
2010 REM ---       Hauptprogramm    ---
2020 REM ------------------------------
2030 :
2040 breite = 4*16+40
2050 grund  = 20
2060 abstand= 20
2070 :
2080 TAG
2090 :
2100 FOR i=1 TO anzahl
2110    untenx=i*abstand+i*breite
2120    unteny=grund
2130    obenx =i*abstand+(i-1)*breite
2140    obeny =grund+hoehe(i,1)
2150    farbe =1
2160    GOSUB 3040
2170 :
2180    MOVE obenx+4,unteny+(obeny-unteny)/2+6
2190    PRINT"Gewinn";
2200 :
2210    unteny=obeny
2220    obeny =unteny+hoehe(i,2)
2230    farbe =2
2240    GOSUB 3040
2250 :
2260    MOVE obenx+4,unteny+(obeny-unteny)/2+6
2270    PRINT"Kosten";
2280 :
2290    unteny=obeny
2300    obeny =unteny+hoehe(i,3)
2310    farbe =3
2320    GOSUB 3040
2330 :
2340    MOVE obenx+4,unteny+(obeny-unteny)/2+6
2350    PRINT"Steuer";
2360 :
2370    MOVE obenx+20,15
```

Programm 4.3: Zweidimensionale Balkendiagramme mit Farbstaffelung (Forts.)

```
2380    PRINT text$(i);
2390 NEXT
2400 :
2410 END
2420 :
3000 REM ------------------------------
3010 REM ---            Block         ---
3020 REM ------------------------------
3030 :
3040    FOR linie=unteny TO obeny
3050       MOVE obenx,linie
3060       DRAWR breite,0,farbe
3070    NEXT
3080 RETURN
3090 :
```

Programm 4.3: Zweidimensionale Balkendiagramme mit Farbstaffelung (Forts.)

Dargestellt werden fünf Balken, wie es in Zeile 1070 dokumentiert ist. Für diese fünf Balken müssen jeweils die drei Werte (wert(,)) die Gesamtbeschriftung (text$()) sowie die Summe aller drei Werte eines einzelnen Balkens, die Höhe der einzelnen Teilbalken und ihr prozentualer Anteil als Variablen vorgehalten werden. Der prozentuale Anteil jedes einzelnen Teilbalkens wurde von uns nicht ausgegeben, dies haben wir für Sie als Übung vorgesehen.

Ab Zeile 1170 werden zunächst die Daten definiert, wobei wir als Beispiel für den Gesamtbalken den Umsatz eines jeden Jahres vorgesehen haben und für die Teilbalken jeweils den Gewinn, die Kosten und die Steuer. Für jedes Jahr ist eine einzelne DATA-Zeile vorgesehen. In den Programmschleifen ab Zeile 1230 werden diese Daten eingelesen; zunächst der Beschriftungstext eines Balkens und dann die drei Werte, die noch in ihrer Höhe normiert werden. Außerdem wird innerhalb der Schleife ab Zeile 1260 die Summe aller Balken errechnet (bereits aus der normierten Höhe). Da der prozentuale Anteil eines jeden Teilbalkens neu errechnet werden kann, wenn die Gesamthöhe vorliegt, ist hierfür eine Extra-Schleife ab Zeile 1320 vorgesehen.

Als Breite des Balkens ist zunächst die Beschriftungsbreite (4 x 16 Bildschirmpunkte für vier Zeichen) plus 40 Zeichen vorgesehen. Die Höhe von der Bildschirmuntergrenze und der Abstand zwischen den einzelnen Balken ist einheitlich mit 20 Bildschirmpunkten vorgesehen.

Als nächstes wird der TAG-Befehl verwendet, da alle folgenden Textausgaben aufgrund der Position des Grafik-Cursors angezeigt werden sollen. Die Programmschleife ab Zeile 2100 wird für alle Balken durchlaufen, wobei wir auf eine Schachtelung der Schleifen verzichtet haben, da einerseits nur drei Teilbalken auszugeben sind, andererseits die Vorbesetzung für den Eintritt in eine geschachtelte Schleife zu umfangreich wäre.

Zunächst wird für jeden Balken die rechte X-Koordinate berechnet, wobei auch der Abstand vom linken Bildschirmrand zum ersten Balken gleich dem Abstand zwischen den Balken ist. Das untere Ende des Balkens wird dann auf den vorbesetzten „grund" gesetzt. Die linke Kante eines Balkens ergibt sich aus der rechten Kante vermindert um die Breite des Balkens, und zum oberen Ende gelangt man, wenn man zum unteren Ende die Höhe (normiert) addiert. Der untere Teilbalken wird in der Farbe 1 gezeichnet, indem das Unterprogramm ab Zeile 3000 aufgerufen wird.

Anschließend erfolgt die Beschriftung des unteren Teils, wobei vom linken Balkenrand um vier Bildschirmpunkte nach rechts gegangen und die Schrift etwas oberhalb der Mitte angesiedelt wird. Mathematiker werden sicherlich die Klammern und die sich aufhebende Wirkung von unteny in Zeile 2180 bemängeln, wir haben dies jedoch im Programmlisting belassen, um Ihnen die Rechenweise darzulegen.

Für den zweiten Balken wird dann ab Zeile 2210 ebenfalls die obere und untere Höhe berechnet, wobei das untere Ende des zweiten Teilbalkens gleich dem oberen Ende des ersten Teilbalkens ist und das obere Ende des zweiten Teilbalkens in der Variablen hoehe(,) festgelegt ist. Dann wird die Farbe umgeschaltet und erneut das Unterprogramm ab Zeile 3000 aufgerufen.

Die Beschriftung erfolgt analog dem ersten Teilbalken. Der dritte Teilbalken wird analog den beiden ersten Teilbalken behandelt.

Als Abschluß erfolgt die Textausgabe der Jahre, wobei wir durch die Vorgabe der Breite eines Balkens (Schriftausgabe + 40 Bildschirmpunkte) zum linken

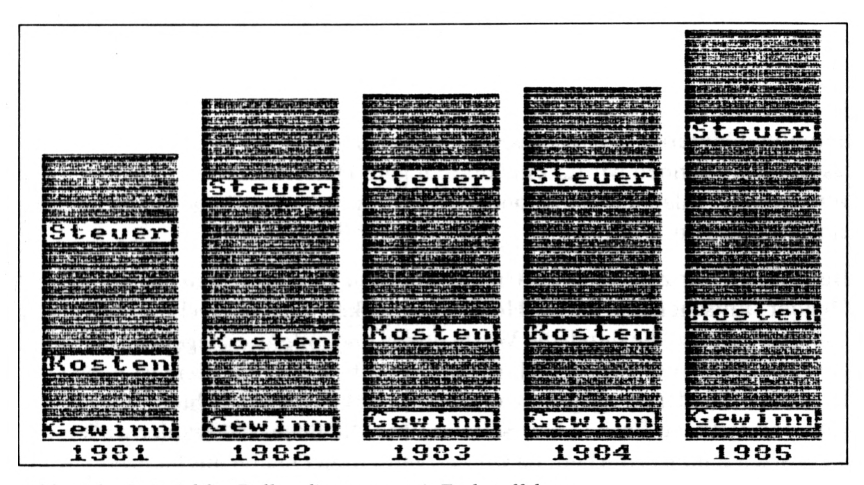


Abb. 4.3: Beispiel für Balkendiagramm mit Farbstaffelung

Rand des Balkens um 20 Bildschirmpunkte nach rechts wandern müssen. Der Wert 15 ist ein Mittel zwischen der Grundzeile, dem unteren Bildschirmrand und der Höhe der Schrift.

Auch hier wieder eine Hardcopy-Ausgabe (Abb. 4.3). Auf eine Variablenübersicht soll hier verzichtet werden, da keine wesentlichen Änderungen gegenüber dem letzten Kapitel vorliegen.

Dreidimensionale Balkendiagramme

Als letztes zum Thema Balkendiagramme wollen wir die dreidimensionalen Balkendiagramme besprechen. Für dreidimensionale Darstellungen sei auch noch auf Kapitel 6 verwiesen, wo das Thema ausführlicher für allgemeine Funktionen behandelt wird.

Dreidimensionale Balkendiagramme könnten auch ähnlich den bisher aufgezeigten Programmen aufgebaut werden, jedoch ginge dann der Überblick verloren, zumal dann, wenn kleinere Balken hinter größeren Balken stehen. Aus diesem Grunde werden zwar die unteren Enden der einzelnen Balken auf eine Grundfläche zusammengefaßt, diese ist jedoch auf dem Bildschirm gedreht dargestellt.

Das folgende Programm ist so allgemein wie möglich gehalten, obwohl wir einige Werte fest eingebaut haben, die jedoch nicht von entscheidender Bedeutung sind. So haben wir das Verhältnis zwischen X- und Y-Koordinaten derart aufgeteilt, daß auf je zwei Bildschirmpunkte in X-Richtung ein Bildschirmpunkt in Y-Richtung versetzt wird. Zur Vereinfachung haben wir eine andere Darstellung der Balken gewählt, als es mit dem Unterprogramm Quader vorgesehen ist (Balken räumlich dargestellt ergeben Quader).

Bei unserem Unterprogramm zum Zeichnen eines Quaders haben wir die vorderen und hinteren Kanten des Quaders parallel zum Bildschirmrand laufen lassen. Dies können wir für eine optisch gute Darstellung im folgenden nicht verwenden, so daß wir ein gesondertes Unterprogramm für gedrehte Quader eingebaut haben.

Besprechen wir zunächst das Unterprogramm zum Zeichnen der gedrehten Quader. Entscheidend für die Höhe eines Balkens sind die rechten und linken äußeren Kanten, die durch die Werte obeny und unteny angegeben sind. Vergleichen Sie hierzu auch die Hardcopy in Abb. 4.4. Außerdem ist noch eine Breite des Quaders vorgegeben, die von der linken bis zur rechten Begrenzung geht und nicht die Breite einer Seite angibt. Mit einem Hilfszähler wollen wir die Verschiebung erreichen, da wir den gedrehten Quader aus mehreren Rauten zusammensetzen wollen.

```
1000 REM ------------------------------
1010 REM ---    Vorspann und Daten   ---
1020 REM ------------------------------
1030 :
1040 BORDER 0
1050 MODE 1
1060 :
1070 anzahlx = 3
1080 anzahly = 5
1090 :
1100 DIM wert (anzahlx,anzahly)
1110 :
1120 DATA 20,50,60
1130 DATA 30,70,80
1140 DATA 40,90,100
1150 DATA 60,120,150
1160 DATA 40,150,180
1170 :
1180 FOR i=1 TO anzahly
1190    FOR j=1 TO anzahlx
1200       READ wert(j,i)
1210    NEXT
1220 NEXT
1230 :
1240 abstandy=320/anzahly
1250 abstandx=320/anzahlx
1260 :
1270 IF abstandx <= abstandy THEN breite=abstandx-6 ELS
E breite=abstandy-6
1280 :
2000 REM ------------------------------
2010 REM --- Zeichnen der 'Grund-   ---
2020 REM ---                platte ---
2030 REM ------------------------------
2040 :
2050 FOR i=0 TO anzahly
2060    MOVE i*abstandy,200-i*abstandy/2
2070    DRAW 320+i*abstandy,360-i*abstandy/2
2080 NEXT
2090 :
2100 FOR i=0 TO anzahlx
2110    MOVE i*abstandx,200+i*abstandx/2
2120    DRAW 320+i*abstandx,40+i*abstandx/2
2130 NEXT
2140 :
3000 REM ------------------------------
3010 REM ---  Zeichnen der Quader    ---
3020 REM ------------------------------
3030 :
3040 FOR i=anzahlx TO 1 STEP -1
3050    x=(i-1)*abstandx+45
3060    y=200+(i-1)*abstandx/2-3
3070    FOR j=1 TO anzahly
3080       unteny=y
3090       obeny=unteny+wert(i,j)
3100       GOSUB 4000
3110       x=x+abstandy
```

Programm 4.4: Dreidimensionale Balkendiagramme

```
3120        y=y-abstandy/2
3130     NEXT
3140 NEXT
3150 :
3160 END
3170 :
4000 REM ------------------------------
4010 REM ---    gedrehter Quader    ---
4020 REM ------------------------------
4030 :
4040 zaehler=0
4050 :
4060 FOR quader=x TO x+breite/2
4070     MOVE quader,obeny-zaehler
4080     DRAW quader,unteny-zaehler,1
4090     zaehler=zaehler+0.5
4100 NEXT
4110 :
4120 FOR quader=x+breite/2 TO x+breite
4130     MOVE quader,obeny-zaehler
4140     DRAW quader,unteny-zaehler,2
4150     zaehler=zaehler-0.5
4160 NEXT
4170 :
4180 zaehler=0
4190 wende=x+breite/2
4200 :
4210 FOR quader=x TO x+breite
4220     MOVE quader,obeny+zaehler
4230     DRAW quader,obeny-zaehler,3
4240     IF quader < wende THEN zaehler=zaehler+0.5 ELSE
 zaehler=zaehler-0.5
4250 NEXT
4260 :
4270 RETURN
```

Programm 4.4: Dreidimensionale Balkendiagramme (Forts.)

In der ersten Programmschleife ab Zeile 4060 wird der linke untere Teil eines Quaders gezeichnet, der vom linken Rand bis zur Mitte geht. Durch den Zähler wird ein Versatz der Linie bei jedem zweiten Bildschirmpunkt in X-Richtung und um einen Bildschirmpunkt in Y-Richtung erzielt. Die Programmschleife ab Zeile 4120 hat nur andere Werte für die Laufvariable (von der Mitte bis zum rechten Rand), für den Zähler (hier negativ) und die Farbe. Dadurch werden die Linien wieder versetzt nach oben gezeichnet. Jetzt haben wir ein Bild, das einem aufgeschlagenen Buch ähnlich sieht, wobei beide Teile verschieden eingefärbt sind.

Zur Vervollständigung setzen wir obendrauf noch eine querliegende Raute, wozu wir aber eine weitere Variable, wende, benötigen, um uns zwei weitere Programmschleifen zu ersparen. In der Programmschleife ab Zeile 4210 wird

dann für die ganze Breite die Raute gezeichnet, wobei der Zähler zunächst anwächst und ab der Mitte wieder zurückgezählt wird. Die Vorgehensweise wird noch deutlicher, wenn Sie das Programm abgetippt haben und sich die Reihenfolge der Darstellung am Rechner ansehen.

Eine räumliche Darstellung von Balken ist dann angezeigt, wenn man – wie bei unserem Beispiel Liniendiagramm – verschiedene Objekte über mehrere Perioden mit den zugehörigen Werten ausgegeben haben möchte.

Statt der Anzahl der Objekte und der Anzahl der Werte je Objekt haben wir uns hier allgemein auf eine Anzahl in X-Richtung und auf eine Anzahl in Y-Richtung (anzahlx, anzahly) festgelegt, die in unserem Beispiel die Werte 3 und 5 annehmen. Entsprechend wird – wie bei den vorigen Programmen auch – eine Matrix wert(,) festgelegt, deren Daten ab Zeile 1120 zu finden sind und in den geschachtelten Programmschleifen ab Zeile 1180 eingelesen werden.

Für spätere Berechnungen werden noch zwei Hilfsvariablen abstandy und abstandx benötigt. Als Grundplatte dient uns eine Raute (siehe Abb. 4.4), die vom linken bis zum rechten Bildschirmrand reicht, aber in der Höhe nicht den ganzen Bildschirm ausfüllt (wegen des Umrechnungsverhältnisses 1:2). Die Raute verläuft also über die Bildschirmpunkte (0,200), (320,360), (639,200) und (320,40). Innerhalb dieser Raute wollen wir nun Parallelogramme einzeichnen, die ein Raster für die einzelnen Balken bilden.

Da die Rasterung natürlich abhängig von der Anzahl jeweils in X- und Y-Richtung ist, muß diese variabel gewählt werden, wozu wir die Variablen abstandy und abstandx verwenden. Außerdem haben wir die Breite der Balken variabel gehalten, wobei der kleinste Abstand zu berücksichtigen ist.

Bei der Grundplatte zeichnen wir zunächst die Linien von unten links nach oben rechts. Ähnlich wie bei den Liniendiagrammen brauchen wir hier auch zur kompletten Umgrenzung eine Linie mehr als Rasterteilungen auszugeben sind, was wir durch Schleifenbeginn mit 0 erreichen. Die X-Richtung ergibt sich aus unserer Hilfsvariablen abstandy, die Y-Richtung ist vom Bildschirmmittelpunkt auszurechnen und halb so groß wie die Abweichung in der X-Richtung.

Damit haben wir den Anfangspunkt der Linien. Der Endpunkt der Linien ist analog zum Anfangspunkt ab der Bildschirmmitte in X-Richtung zu errechnen und für die Y-Richtung entsprechend vom obersten Punkt der Grundraute.

Analog erfolgt die Ausgabe der Raster quer zu den bisher gezeichneten. Hierbei ist jedoch unsere Hilfsvariable abstandx zu verwenden und für die Positionierung des Grafik-Cursors am Anfang der Linie bei der Y-Koordinate im oberen Bildschirmbereich anzusetzen. Der Endpunkt der Linie befindet sich

im rechten unteren Bildschirmviertel; dementsprechend sind die Faktoren 320 bei der X-Koordinate und 40 bei der Y-Koordinate zu berücksichtigen.

Etwas Rechnerei braucht man nun, um die Quader innerhalb dieses Rasters zu positionieren. Dabei wollen wir die äußere Schleife in Y-Richtung laufen lassen,wobei wir die Quader – optisch gesehen – von hinten nach vorne zeichnen, um eine korrekte Überlappung zu erreichen. Für jede Reihe von Balken in Y-Richtung werden zunächst die Punkte für den ersten Balken in den Zeilen 3050 und 3060 festgelegt. Für jeden weiteren Balken werden sie in den Zeilen 3110 und 3120 verändert. Dazwischen erfolgt nur die Festlegung des oberen und unteren Endes des Balkens und der Aufruf zum Zeichnen der Darstellung.

Auch hier zeigen wir wieder eine Hardcopy zu den dargestellten Daten (Abb. 4.4).

4.3 TORTEN-, KREIS- UND SÄULENDIAGRAMME

Wir haben mit den mehrstufigen Balkendiagrammen eine Möglichkeit kennengelernt, Aufteilungen einer Grundgesamtheit zu kennzeichnen. Bei drei, vier oder fünf Teilen ist dies – je nach Länge eines Balkens – auch noch übersichtlich. Eine bessere und allgemeinere Möglichkeit, eine Grundgesamtheit im Einzelfall in allgemeine Segmente aufzuteilen, bieten die Kreis- bzw. Tortendiagramme. Hier werden innerhalb eines Kreises die verschiedenen Prozentzahlen durch entsprechend große Kreissegmente dargestellt. Da wir bis-

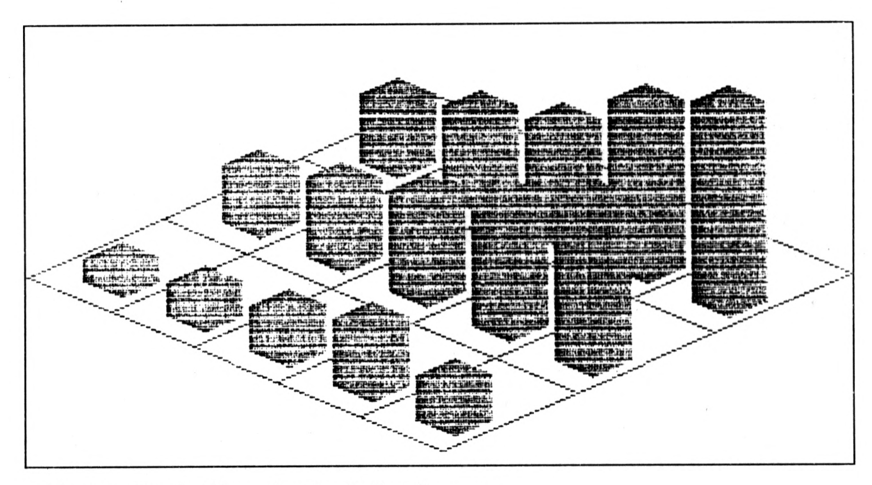


Abb. 4.4: Ein dreidimensionales Balkendiagramm

her die BASIC-Unterprogramme verwendet haben, wollen wir an dieser Stelle die Befehlserweiterungen heranziehen.

Weil wir das Erfassen der Zahlen, die grafisch dargestellt werden sollen, bereits in den beiden vorangegangenen Kapiteln ausführlich vorgestellt haben, wollen wir uns hier auf die Eingabe innerhalb von DATA-Zeilen beschränken. Außerdem werden noch die Beschriftungen weggelassen, da auch diese Thematik bereits behandelt wurde, und weiterhin wurde auf einen Hardcopy-Ausdruck verzichtet, da die verschiedenen Farben der Sektoren nicht zur Geltung kämen. Hier zunächst das Listing für alle drei Diagrammformen:

```
1000 '---------------------
1010 '--- Kreisdiagramm ---
1020 '---------------------
1030 :
1040 MODE 1
1050 :
1060 DATA 5,155,1,134,2,348,3,293,1,215,2
1070 :
1080 RESTORE 1060
1090 mittex  = 320
1100 mittey  = 200
1110 radiusx = 200
1120 radiusy = 100
1130 :
1140 GOSUB 5000
1150 :
1160 CALL &BB18 : CLS
1170 :
2000 '-----------------------
2010 '--- Saeulendiagramm ---
2020 '-----------------------
2030 :
2040 mittex  = 100
2050 mittey  = 150
2060 radiusx =  50
2070 radiusy =  20
2080 hoehe   = 130
2090 farbe1  =   3
2100 farbe2  =   1
2110 :
2120 GOSUB 6000
2130 :
2140 CALL &BB18 : CLS
2150 :
3000 '----------------------
3010 '--- Tortendiagramm ---
3020 '----------------------
3030 :
3040 DATA 3,20,1,50,2,30,3
3050 :
3060 RESTORE 3040
```

Programm 4.5: Kreisdiagramme, Säulendiagramme, Tortendiagramme

```
3070 mittex  = 200
3080 mittey  =  80
3090 radiusx = 150
3100 radiusy =  50
3110 hoehe   = 100
3120 farbel  =   1
3130 farbe2  =   2
3140 :
3150 GOSUB 7000
3160 :
3170 CALL &BB18 : CLS
3180 :
4998 END
4999 :
5000 '-------------------------
5010 '--- Unterprogramm fuer ---
5020 '--- Kreisdiagramme     ---
5030 '--- Die Anzahl und die ---
5040 '--- Groesse der Sektoren -
5050 '--- wird mit einer DATA- -
5060 '--- Zeile  uebergeben. ---
5070 '-------------------------
5080 :
5090 READ anzahl
5100 DIM farbe(anzahl) , sektor(anzahl)
5110 gesamt = 0
5120 :
5130 FOR i=1 TO anzahl
5140     READ sektor(i) , farbe(i)
5150     gesamt = gesamt + sektor(i)
5160 NEXT
5170 :
5180 FOR i=1 TO anzahl
5190     sektor(i) = 360 * sektor(i) / gesamt
5200 NEXT
5210 :
5220 gesamt=0
5230 :
5240 FOR i=1 TO anzahl
5250      FOR winkel=gesamt TO gesamt+sektor(i)
5260         |RADIUS,mittex,mittey,radiusx,radiusy,wink
el,farbe(i)
5270     NEXT
5280     gesamt = gesamt + sektor(i)
5290 NEXT
5300 :
5310 |KREIS,mittex,mittey,radiusx,radiusy,farbe(1)
5320 :
5330 ERASE farbe,sektor
5340 :
5350 RETURN
5360 :
6000 '-------------------------
6010 '--- Unterprogramm fuer ---
6020 '--- Saeulendiagramme   ---
6030 '-------------------------
6040 :
```

Programm 4.5: Kreisdiagramme, Säulendiagramme, Tortendiagramme (Forts.)

```
6050 |BLOCK,mittex-radiusx,mittey,mittex+radiusx,mittey
+hoehe,farbe1
6060 |SCHEIBE,mittex,mittey,radiusx,radiusy,farbe1
6070 |SCHEIBE,mittex,mittey+hoehe,radiusx,radiusy,farbe
2
6080 |BOGEN,mittex,mittey,radiusx,radiusy,180,361,2,far
be2
6090 MOVE mittex-radiusx,mittey : DRAWR 0,hoehe,farbe2
6100 MOVE mittex+radiusx,mittey : DRAWR 0,hoehe,farbe2
6110 :
6120 RETURN
6130 :
7000 '-------------------------
7010 '--- Unterprogramm fuer ---
7020 '--- Torten            ---
7030 '-------------------------
7040 :
7050 |BLOCK,mittex-radiusx,mittey,mittex+radiusx,mittey
+hoehe,farbe1
7060 |SCHEIBE,mittex,mittey,radiusx,radiusy,farbe1
7070 |BOGEN,mittex,mittey,radiusx,radiusy,180,361,5,far
be2
7080 MOVE mittex-radiusx,mittey : DRAWR 0,hoehe,farbe2
7090 MOVE mittex+radiusx,mittey : DRAWR 0,hoehe,farbe2
7100 :
7110 mittey = mittey + hoehe
7120 :
7130 GOSUB 5000
7140 :
7150 |KREIS,mittex,mittey,radiusx,radiusy,farbe2
7160 :
7170 mittey = mittey - hoehe
7180 :
7190 RETURN
7200 :
```

Programm 4.5: Kreisdiagramme, Säulendiagramme, Tortendiagramme (Forts.)

Zunächst wird der Bildschirm auf den Vier-Farben-Modus eingestellt, dann werden die Daten festgelegt, wobei die erste Zahl die Anzahl der Aufteilungen angibt und die nächsten Zahlen jeweils Gruppen zu je zwei Zahlen bilden, wovon die erste Zahl den darzustellenden Wert und die zweite Zahl die Farbe angibt. Der darzustellende Wert kann in seiner ursprünglichen Form eingegeben werden, da er später noch entsprechend normiert wird.

Kreisdiagramm

Durch den RESTORE-Befehl ist eine Auswahl zwischen verschiedenen DATA-Zeilen möglich. Ab Zeile 1090 werden die Lage und die Größe des Kreises (der Ellipse) festgelegt, und dann wird das Unterprogramm zum Zeichnen des Kreises aufgerufen. Der CALL-Befehl in Zeile 1160 wartet nur auf Tasten-

druck und soll Ihnen eine weitere Möglichkeit gegenüber der Fassung mit INKEY$ zeigen. Für das nächste Diagramm wird dann noch der Bildschirm gelöscht.

Säulendiagramm

Auch bei dem Säulendiagramm werden die Lage und die Größe festgelegt, wobei hier zusätzlich zu den beim Kreis benötigten Daten noch die Höhe und die einzelnen Farben angegeben werden müssen. Auch hier wird das entsprechende und weiter unten beschriebene Unterprogramm aufgerufen und anschließend wieder auf einen Tastendruck gewartet.

Tortendiagramm

Das Tortendiagramm ist eine Verbindung zwischen Kreisdiagramm und Säulendiagramm, d. h. dem Kreisdiagramm wird noch eine Säule angehängt bzw. bei einer Säule wird die obere Fläche entsprechend den Angaben aufgeteilt. Auch in den Programmzeilen ab 3000 werden lediglich die Parameter für das Unterprogramm vorbesetzt.

Unterprogramm zum Kreisdiagramm

Ab Zeile 5090 werden zunächst die Daten, die durch den RESTORE-Befehl in Zeile 1080 bestimmt sind, eingelesen, und die Summe aller Werte wird bestimmt. Die Normierung erfolgt in der Programmschleife ab Zeile 5180. Dann werden in zwei ineinandergeschachtelten Schleifen die einzelnen Radien ausgegeben, wobei eine annähernd ausgefüllte Fläche bei den einzelnen Sektoren entsteht. Sollen die Flächen nicht ausgefüllt werden, so ist die FOR...NEXT-Schleife in den Zeilen 5250 bis 5270 durch die einfache Ausgabe eines einzigen Radius zu ersetzen. Abschließend wird in Zeile 5310 noch ein Umkreis um die Darstellung gezogen, und die Felder farbe und sektor werden gelöscht, damit sie für weitere Ausgaben neu definiert werden können.

Unterprogramm für Säulendiagramme

Der erste Gedanke, wie man ein Säulendiagramm erstellen kann, wird wahrscheinlich das Aneinanderreihen von Halbkreisen (Halbellipsen) für die Säule sein. Schneller geht es, wenn man die Säule aus einem Block und zwei Scheiben bildet. Also wird auch im Unterprogramm ab Zeile 6050 zunächst der Block mit den entsprechenden Parametern ausgegeben. In den Zeilen 6060 und 6070 werden die obere und die untere Scheibe dargestellt, in Zeile 6080 die untere Begrenzung der Säulen und in den Zeilen 6090 und 6100 die seitlichen Begrenzungen der Säulen gezeichnet. Bei der Farbwahl in unserem Bei-

spiel ergibt sich also eine orange Säule mit gelbem Deckel und gelber Umrahmung. Diese Säulen lassen sich ähnlich den Balkendiagrammen verwenden.

Unterprogramm für Tortendiagramme

Als letzte Diagrammart wollen wir eine Kombination zwischen Kreisdiagramm und Tortendiagrammen darstellen. Wie bei den Säulen werden hier auch zunächst der Block und die untere Begrenzungsscheibe gezeichnet sowie die untere und seitliche Umrahmung. Nach entsprechender Umbesetzung des Parameters mittey kann das Unterprogramm zum Zeichnen eines Kreisdiagramms aufgerufen werden, um die obere Fläche entsprechend in Sektoren aufzuteilen.

Es können also mehrere Gesamtwerte durch die Säulenhöhe bestimmt werden, wobei diese Gesamtwerte am oberen Teil der Säule nochmals aufgegliedert werden können.

Mit den hier vorgestellten Diagrammarten sollte es Ihnen nicht schwerfallen, die passende Darstellung je nach gewünschtem Anwendungsfall auszuwählen.

Kapitel 5

Künstlerische Grafiken und allgemeine Gestaltungsmöglichkeiten

Das folgende Kapitel ist ganz der Optik am Bildschirm gewidmet. Wir wollen verschiedene Verfahren vorstellen, mit denen Sie hübsche Grafiken entwerfen können, sei es, um sie nur zu betrachten, um sie als Titelbild für eigene Videofilme zu benutzen, als Vorspann eines Programms oder für andere Zwecke.

Zunächst wollen wir im ersten Teil für die mathematisch nicht versierten Leser die beiden wichtigen Winkelfunktionen Sinus und Cosinus anhand von Beispielen eingehend erklären, da sie als Grundlage für alle aus Kurven bestehenden Grafiken dienen.

Im weiteren wollen wir die maximal möglichen sechzehn Farben ausnutzen und diese bei speziell dafür entworfenen Grafiken gegeneinander austauschen, so daß der Eindruck von laufenden Farben entsteht.

Der nächste Teil ist dem Transparent-Modus gewidmet, womit Sie z. B. Ihre Gafiken unsichtbar ausgeben können, um anschließend durch eine Farbumschaltung diese auf einen Schlag sichtbar zu machen.

Der letzte Teil widmet sich besonderen Tricks, wobei die Möglichkeiten des Rechners weiter ausgereizt werden.

5.1 EINFÜHRUNG

Wie bereits erwähnt, wollen wir zunächst auf die beiden Winkelfunktionen Sinus und Cosinus eingehen, ohne diese jedoch mathematisch herzuleiten. Anhand von einigen Beispielen wollen wir ihre Wirkungsweise erläutern und die Möglichkeiten der Beeinflussung prüfen.

Die Winkelfunktionen – auch trigonometrische Funktionen genannt – sind in

Ihrem Rechner als Funktionen enthalten (SIN(Argument); COS(Argument)). „Argument" ist dabei ein beliebiger Zahlenwert, der vom Rechner selbst in den zulässigen Bereich gebracht wird. Wir werden im weiteren nur mit positiven Argumenten arbeiten.

Betrachten wir zunächst die Sinus-Funktion. Das Ergebnis eines Aufrufs von SIN() ist ein Wert zwischen 0 und 1. Diese Werte werden periodisch (ein sich stetig wiederholender Bereich) angenommen, wobei eine Periode 2*PI (Kreisumfang am Kreis mit dem Radius 1) beträgt. D. h. die Werte ergeben sich ähnlich der Modulo-Funktion, d. h. man kann beliebig oft 2*PI vom Argument abziehen und erhält immer noch den gleichen Wert. Soweit zur periodischen Eigenschaft der Winkelfunktion, nun zu einer weiteren Besonderheit: Bei den Werten 0, PI und 2*PI ist der Wert der Sinus-Funktion 0. Bei 0,5*PI ist der Wert 1 und bei 1,5*PI ist der Wert −1.

Da sich mit den Schneider-Computern auch Berechnungen im Gradmaß durchführen lassen, sind als Argumente natürlich auch Gradzahlen zulässig, wobei sich hier die Werte selbstverständlich alle 360 Grad wiederholen. Der Rechner muß vorher natürlich den DEG-Befehl erhalten. Hier eine kurze Tabelle von Gradzahlen und den entsprechenden Werten der Sinus-Funktion:

Gradzahl	Sinus-Funktion
30	0,5
90	1
150	0,5
180	0
210	−0,5
270	−1
330	−0,5
360	0

Sofern Sie mit Werten von PI nicht so vertraut sind, hilft die Rückführung auf den sogenannten Einheitskreis, d. h. einen Kreis mit dem Radius 1. Der Umfang eines Kreises ist bekanntlich 2*PI*Radius. Da der Radius am Einheitskreis 1 ist, ergibt sich für den Umfang 2*PI und somit für die Hälfte des Kreises PI.

Nun genug der Vorrede, beginnen wir mit dem ersten Beispielprogramm (Programm 5.1).

Nach einigen Vorarbeiten wie Einstellen des Bildschirm-Modus (bei gleichzeitigem Löschen des Bildschirms) und Vorbesetzen der Farbauswahlnummern 0 und 1 sowie dem Ziehen einer waagerechten Linie in der Bildschirmmitte

```
100 REM ------------------------------
110 REM ---     Sinuskurve         ---
120 REM ------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 0,200
220 :
230 FOR i=0 TO 2*PI STEP 0.01
240    DRAW x,200+180*SIN(i),1
250    x=x+1
260 NEXT
270 :
```

Programm 5.1: Zeichnen einer Sinuskurve

und Positionierung des Cursors am Anfang dieser Linie werden in einer Schleife von 0 bis 2∗PI mit der im Normalfall recht kleinen Schrittweite von 0,01 fortlaufend die Werte der Sinus-Funktion ausgegeben.

Da wir schon gesagt haben, daß das Ergebnis der Sinus-Funktion zwischen 0 und 1 liegt, multiplizieren wir es mit 180, weil wir es sonst auf dem Bildschirm nicht erkennen können. Wenn das Ergebnis der Sinus-Funktion den Wert 1

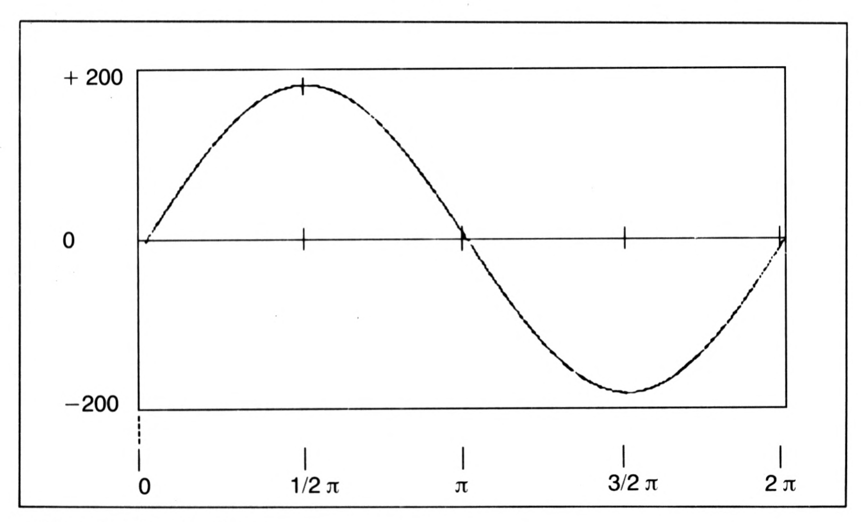


Abb. 5.1: Eine Sinuskurve

annimmt, ergibt sich ein Abstand von der Mittelachse von 180 Bildschirmpunkten. Die Mittelachse am Bildschirm ist als Konstante (200) im DRAW-Befehl angegeben.

Damit die Werte nicht immer auf der gleichen X-Koordinate auf- und abwärts eingetragen werden, lassen wir für jede neue Errechnung der Sinus-Funktion den X-Wert um 1 erhöhen. Beachten Sie bitte, daß die Sinus-Funktion auch negative Werte annehmen kann (maximal −1) und somit auch eine Y-Koordinate von 20 (200−180) möglich ist. Das Ergebnis sehen Sie in Abb. 5.1.

Diese Kurve ist Ihnen sicherlich schon aus anderen Darstellungen bekannt. Nun können Sie selbst solche Kurven erzeugen.

Im weiteren wollen wir zunächst die Kurve seitlich stauchen und wieder in die Länge ziehen. Im nächsten Programm haben wir lediglich die Angaben in der FOR...NEXT-Schleife geändert (Programm 5.2).

In Zeile 230 wurde zunächst der Wert des Schleifenendekriteriums verdoppelt, so daß quasi zwei „Vollkreise" durchlaufen werden. Dies würde bei einer Schrittweite wie in Abb. 5.1 optisch nicht sehr viel bringen, da der Bildschirmrand schon erreicht ist. Deshalb wurde die Schrittweite verdoppelt, d. h. im Gegensatz zum vorher dargestellten Listing wird nun jeder zweite Wert errechnet. Die kleinen Schrittweiten sind natürlich selbstverständlich, da wir über 600 Bildschirmpunkte füllen wollen, der Wert von 4*PI jedoch nur etwa 12,5 beträgt. Bei der normalerweise vorgegebenen Schrittweite 1 würden wir also nur 12 Bildschirmpunkte erhalten, die man mit der Lupe suchen müßte.

Das Ergebnis der neuen Listings finden Sie in Abb. 5.2.

```
100 REM --------------------------------
110 REM ---     Sinuskurve           ---
120 REM --------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 0,200
220 :
230 FOR i=0 TO 4*PI STEP 0.02
240    DRAW x,200+180*SIN(i),1
250    x=x+1
260 NEXT
```

Programm 5.2: Geändertes Sinus-Programm

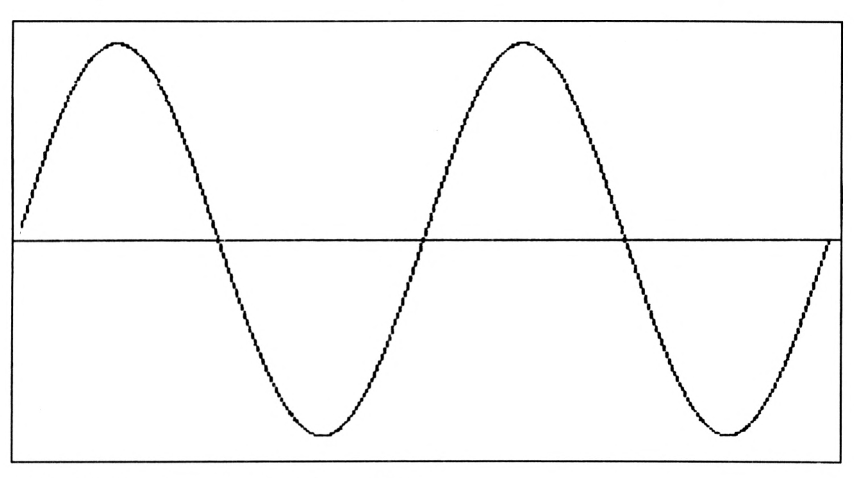


Abb. 5.2: Sinuskurve im Bereich von 0 bis 6/π

Nun wollen wir das Ganze wieder strecken, jedoch ohne daß wir die Schrittweite verändern (Programm 5.3).

Wir vergrößern hier lediglich die Verschiebung auf der X-Achse (siehe Zeile 250). Der Unterschied ist optisch fast nicht sichtbar, wie Abb. 5.3 beweist.

Die Ausdehnung der Sinus-Kurve zu den Seiten hin können wir also einerseits durch die Wahl des Abstands auf der X-Achse beeinflussen, andererseits durch die Wahl der Schrittweite für die Schleife zur Berechnung der einzelnen

```
100 REM ------------------------------
110 REM ---      Sinuskurve        ---
120 REM ------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 0,200
220 :
230 FOR i=0 TO 4*PI STEP 0.02
240    DRAW x,200+180*SIN(i),1
250    x=x+2
260 NEXT
```

Programm 5.3: Sinusdarstellung, dritte Möglichkeit

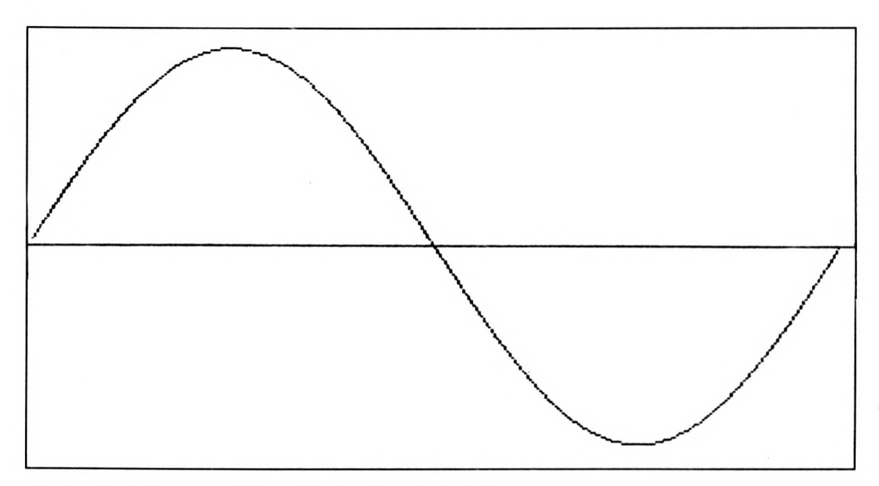


Abb. 5.3: Sinus, gezeichnet durch Programm 5.3

Sinus-Werte. Den Nullpunkt (Beginn) der Kurve können Sie selbst bestimmen, indem Sie in Zeile 210 einen entsprechenden anderen Wert eingeben.

Wie bereits mehrfach in diesem Buch erwähnt, sei auch hier angeraten, mit den Beispielprogrammen etwas herumzuprobieren und die verschiedenen Parameter zu verändern.

Im Vorspann haben wir bereits eine zweite Funktion erwähnt: den Cosinus. Im folgenden Listing haben wir das zweite Beispielprogramm so erweitert, daß die Cosinus-Funktion mit ihren entsprechenden Werten zusätzlich ausgegeben wird (Programm 5.4).

Bringen Sie nun die beiden Kurven nicht durcheinander. Die Sinus-Kurve beginnt auf der rechten Seite an der Mittelachse. Wenn Sie sich nun den Verlauf der beiden Kurven zueinander genau betrachten, werden Sie einige Besonderheiten feststellen. Zu Beginn (Funktionswert = 0) hat die Cosinus-Funktion den Wert 1. Sobald die Sinus-Funktion den Wert 1 annimmt, wird die Cosinus-Funktion 0. Weiter geht es mit SIN=0/COS=−1, SIN=−1/COS=0, und das ganze Spiel beginnt wieder von vorne.

Wir merken uns also, daß beim Sinus-Wert 0 der Cosinus entweder den Wert 1 oder −1 hat und bei Cosinus-Wert 0 der Sinus entweder 1 oder −1 ist. Es gibt sogar mathematische Umrechnungsformeln, die diesen Zusammenhang nutzen, mit denen wir uns aber hier nicht weiter beschäftigen wollen.

Manche werden sich vielleicht jetzt gefragt haben, warum die beiden Funktionen nicht gleichzeitig in einer Schleife gedruckt werden. Dies wäre möglich,

```
100 REM -------------------------------
110 REM --- Sinus- und Cosinuskurve ---
120 REM -------------------------------
130 :
135 INK 2,6
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 0,200
220 :
230 FOR i=0 TO 4*PI STEP 0.02
240    DRAW x,200+180*SIN(i),1
250    x=x+1
260 NEXT
270 :
280 MOVE 0,200
290 x=0
300 :
310 FOR i=0 TO 4*PI STEP 0.02
320    DRAW x,200+180*COS(i),2
330    x=x+1
```

Programm 5.4: Sinus und Cosinus

wenn man statt des DRAW-Befehls den PLOT-Befehl zum Zeichnen verwenden würde. Sofern man jedoch nur eine Schleife verwendet und den DRAW-Befehl benutzt, erhält man Programm 5.5.

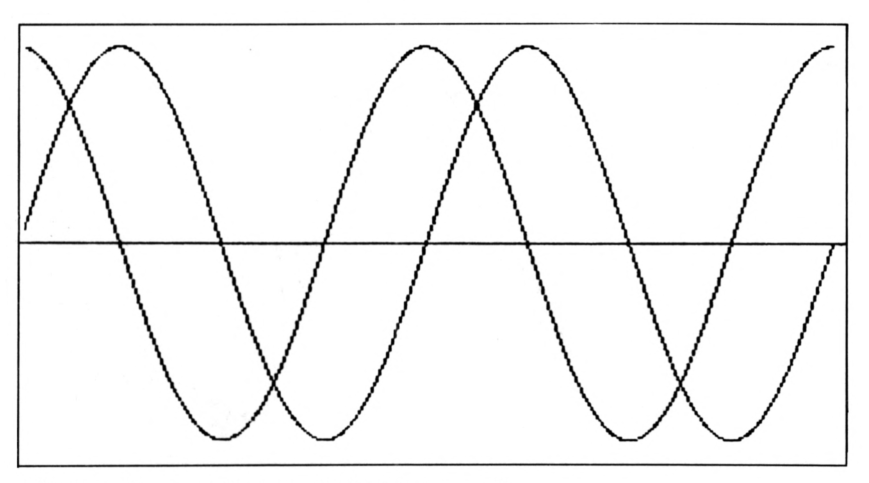


Abb. 5.4: Sinus und Cosinus, gleichzeitig dargestellt

```
100 REM ------------------------------
110 REM --- Sinus- und Cosinuskurve ---
120 REM ------------------------------
130 :
135 INK 2,6
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 0,200
220 :
230 FOR i=0 TO 4*PI STEP 0.02
240    DRAW x,200+180*SIN(i),1
245    DRAW x,200+180*COS(i),2
250    x=x+1
260 NEXT
```

Programm 5.5: Sinus und Cosinus in einer Schleife

Da der Grafik-Cursor immer dort steht, wo er zuletzt verwendet wurde, ergibt sich ein *zweifarbiges* Bild, das in Abb. 5.5 nur unvollkommen wiedergegeben werden kann.

Im folgenden wollen wir die Sinus-Kurve und die Cosinus-Kurve wieder in zwei getrennten Schleifen darstellen und noch eine Summenfunktion bilden. Dazu müssen wir allerdings den Multiplikator für die ermittelten Funktions-

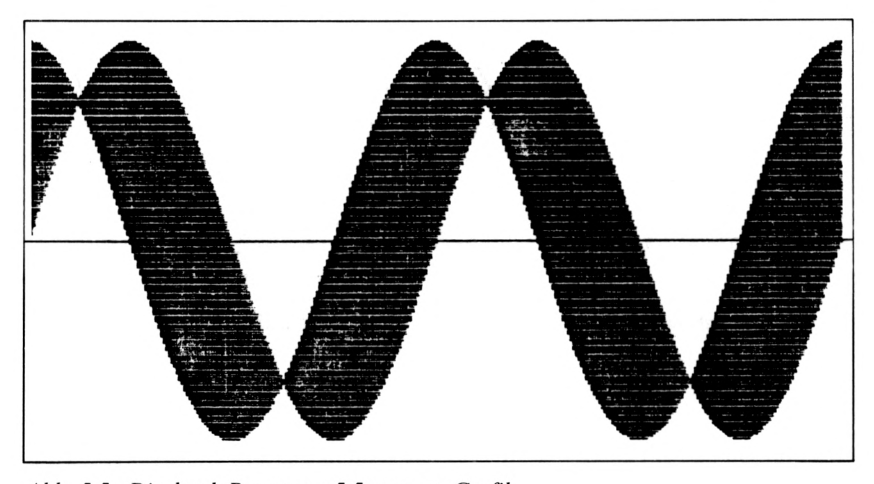


Abb. 5.5: Die durch Programm 5.5 erzeugte Grafik

werte halbieren, um innerhalb des Bildschirmbereichs zu bleiben. Außerdem verwenden wir jetzt drei Farben, die in den Zeilen 160 bis 190 eingetragen werden. Achten Sie also besonders auf die Zeilen 260 und 350. Neu sind die Zeilen 280 und 370 sowie die Zeilen ab 400. Damit wollen wir Ihnen eine Möglichkeit aufzeigen, die Ihnen die Berechnung der Endwerte abnimmt. Sollten Sie den rechten Bildschirmrand erreicht haben, fahren Sie einfach mit der weiteren Programmschleife fort.

Das Ergebnis ist in Abb. 5.6 dargestellt.

Gehen wir wieder ausschließlich zur Sinus-Kurve zurück. Unser zweites Beispielprogramm ändern wir dahingehend ab, daß wir die Werte der Sinus-Funktion quadrieren (Programm 5.7).

```
100 REM -------------------------------
110 REM --- Sinus, Cosinus, Summe   ---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 2,6
180 INK 3,26
190 INK 0,0
200 :
210 MOVE 0,200
220 DRAW 639,200,1
230 MOVE 0,200
240 :
250 FOR i=0 TO 6*PI STEP 0.02
260    DRAW x,200+90*SIN(i),1
270    x=x+1
280    IF x=640 GOTO 310
290 NEXT
300 :
310 MOVE 0,200
320 x=0
330 :
340 FOR i=0 TO 6*PI STEP 0.02
350    DRAW x,200+90*COS(i),2
360    x=x+1
370    IF x=640 GOTO 400
380 NEXT
390 :
400 MOVE 0,200
410 x=0
420 :
430 FOR i=0 TO 6*PI STEP 0.02
440    DRAW x,200+90*SIN(i)+90*COS(i),3
450    x=x+1
460 NEXT
```

Programm 5.6: Sinus, Cosinus und die Summenfunktion

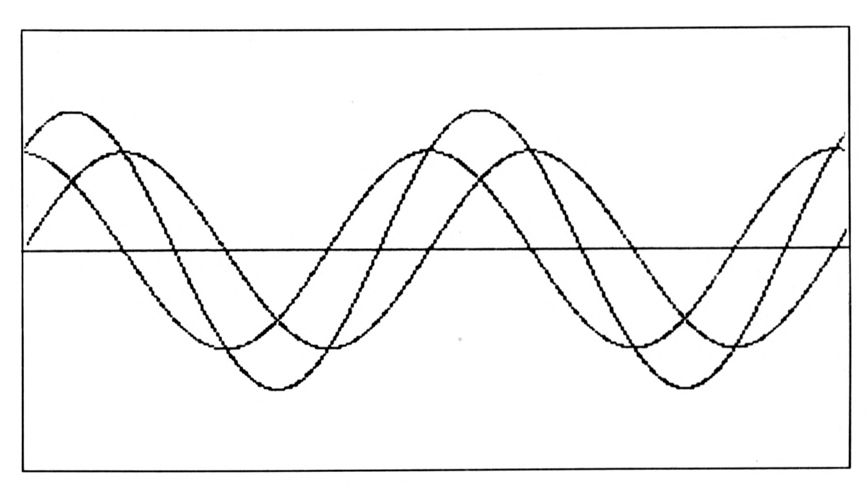


Abb. 5.6: Darstellung von Sinus, Cosinus und Summenfunktion

Wie Ihnen aus der Schulzeit sicherlich noch bekannt ist, ist das Ergebnis einer Quadrierung immer positiv, so daß auch in der Abbildung keine Punkte unterhalb der Mittelachse gezeichnet werden (siehe Abb. 5.7).

Es gibt noch eine weitere trigonometrische Funktion, die in Ihrem Rechner implementiert ist: Tangens, der mit TAN (Argument) aufgerufen wird. Im folgenden haben wir unser altbekanntes Beispiel in Zeile 240 dahingehend geändert, daß der Tangens aufgerufen wird. Da der Tangens jedoch nicht nur Wer-

```
100 REM -------------------------------
110 REM ---     Sinuskurve          ---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 0,200
220 :
230 FOR i=0 TO 4*PI STEP 0.02
240    DRAW x,200+180*SIN(i)^2,1
250    x=x+1
260 NEXT
```

Programm 5.7: Quadrierte Sinus-Funktion

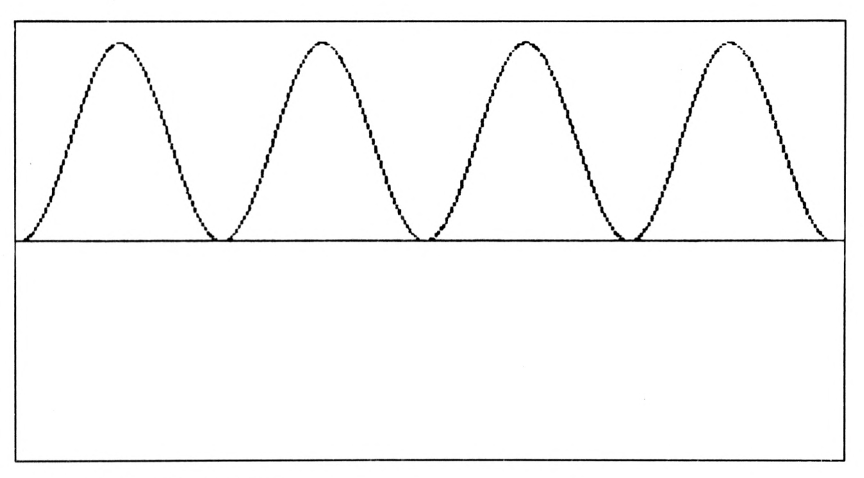


Abb. 5.7: Quadrierter Sinus

te zwischen 0 und 1 annimmt, sondern zwischen 0 und Unendlich, muß man mit dem Multiplikator etwas vorsichtiger sein.

Würde man keinen Multiplikator für die Funktion wählen, so wäre kaum etwas zu sehen. Auch bei der Schrittweite muß man etwas vorsichtiger sein, wir haben 0,005 gewählt. Die Werte des Tangens können Sie sich mit einem einfachen PRINT-Befehl ausgeben lassen, was vielleicht auch bei Sinus und Cosinus zur Verdeutlichung sinnvoll ist.

```
100 REM --------------------------------
110 REM ---         Tangens          ---
120 REM --------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 0,200
220 :
230 FOR i=0 TO 3*PI STEP 0.005
240    DRAW x,200+5*TAN(i),1
250    x=x+1
260 NEXT
```

Programm 5.8: Zeichnen der Tangens-Funktion

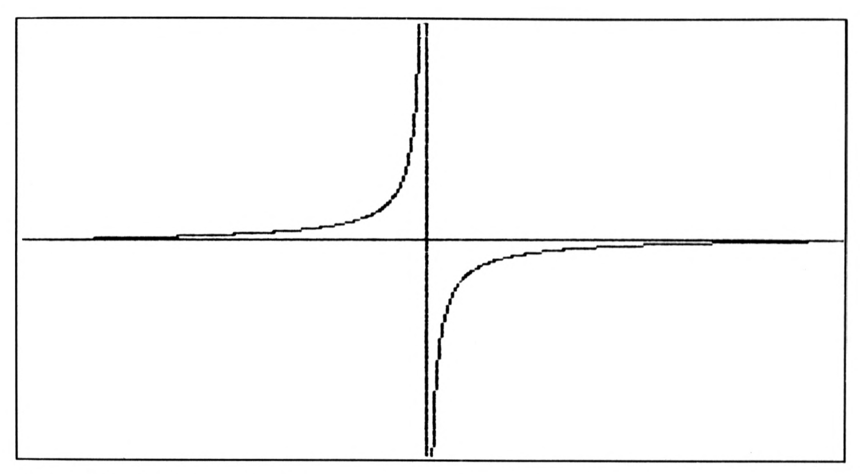


Abb. 5.8: Die Tangens-Funktion

Wie Sie in Abb. 5.8 bei genauerem Hinsehen feststellen, befindet sich ein kleiner Abstand zwischen der Mittelsenkrechten – auch durch den Tagens gezeichnet – und dem Rest der Funktion, der darauf hindeutet, daß an diesen Stellen die Spitze der Funktion ins Unendliche wächst.

```
100 REM -------------------------------
110 REM ---     Quadratfunktion     ---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 ORIGIN 320,200
170 :
180 INK 1,15
190 INK 0,0
200 :
210 MOVE 0,0
220 DRAW 0,200,1
230 DRAW 0,-200,1
240 MOVE 0,0
250 DRAW 320,0,1
260 DRAW -320,0,1
270 :
280 FOR x=-320 TO 320
290    DRAW x,(x/15)^2
300    x=x+1
310 NEXT
```

Programm 5.9: Zeichnen der Quadrat-Funktion

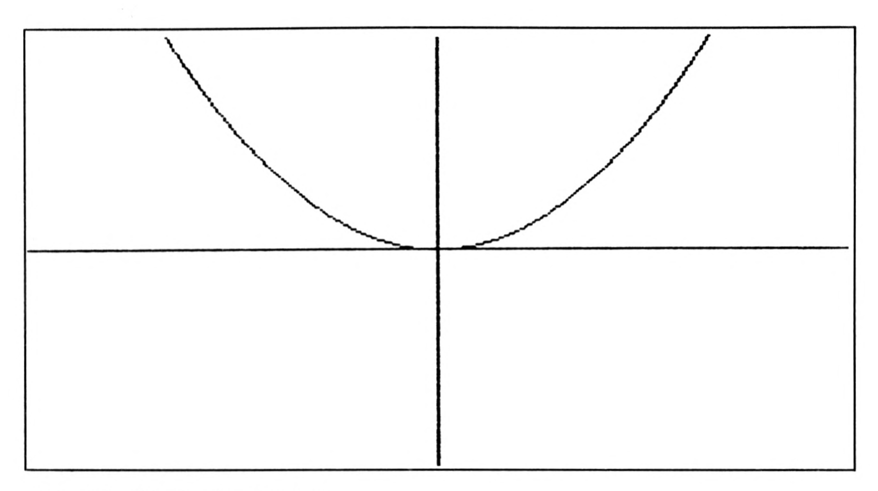


Abb. 5.9: Die Quadrat-Funktion

Bevor wir nun zur Darstellung eines Kreises und weiteren – komplexeren – Beispielen kommen, wollen wir noch zwei recht einfache Funktionen vorstellen, die Quadrat-Funktion und die Kubik-Funktion, stellvertretend für die Gruppe der Exponential-Funktionen. Zunächst die Quadrat-Funktion (Programm 5.9).

In diesem Falle haben wir im Vorspann den Mittelpunkt des Koordinatensystems auf den Mittelpunkt des Bildschirms verlegt (Zeile 160), wodurch wir das übliche karthesische Koordinatensystem erhalten. Dies ist auch bei den vorher beschriebenen Beispielprogrammen möglich, jedoch haben wir es aus Gründen der Einfachheit ausgelassen.

Im weiteren haben wir ab Zeile 210 die Achsen in das Koordinatensystem eingezeichnet. Die Programmschleife muß dann natürlich von –320 bis 320 laufen. Um einen größeren Teil der Quadrat-Funktion darstellen zu können, haben wir den X-Wert noch durch 15 dividiert. Der Akzent vor der abschließenden 2 in Zeile 290 stellt den Potenzierungsoperator dar, der auf der Tastatur durch einen Pfeil nach oben symbolisiert ist. Abb. 5.9 zeigt die zugehörige Hardcopy. Bei Potenzierungen mit 4, 6, 8 ... ändert sich nur die Breite der sogenannten Parabel.

Das Listing für die Kubik-Funktion zeigt Programm 5.10.

Das Ergebnis sehen Sie in Abb. 5.10.

Wie bereits erwähnt, können bei geraden Potenzen nur positive Werte auftreten, so daß alle Funktionswerte in Abb. 5.9 oberhalb der Mittelachse liegen.

```
100 REM -------------------------------
110 REM ---      Kubikfunktion      ---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 ORIGIN 320,200
170 :
180 INK 1,15
190 INK 0,0
200 :
210 MOVE 0,0
220 DRAW 0,200,1
230 DRAW 0,-200,1
240 MOVE 0,0
250 DRAW 320,0,1
260 DRAW -320,0,1
270 :
280 FOR x=-320 TO 320
290    DRAW x,(x/20)^3,1
300    x=x+1
310 NEXT
320 :
330 ORIGIN 0,0
```

Programm 5.10: Die Kubik-Funktion

Dies ist bei ungeraden Potenzen anders. Hier ist die Kurve links von der Y-Achse im negativen Bereich und rechts davon im positiven Bereich. Durch Addition von konstanten Werten jedoch können Sie diese Kurve nach oben oder unten verschieben. Die Ausdehnung in der Breite wird von der Höhe der Potenz bestimmt.

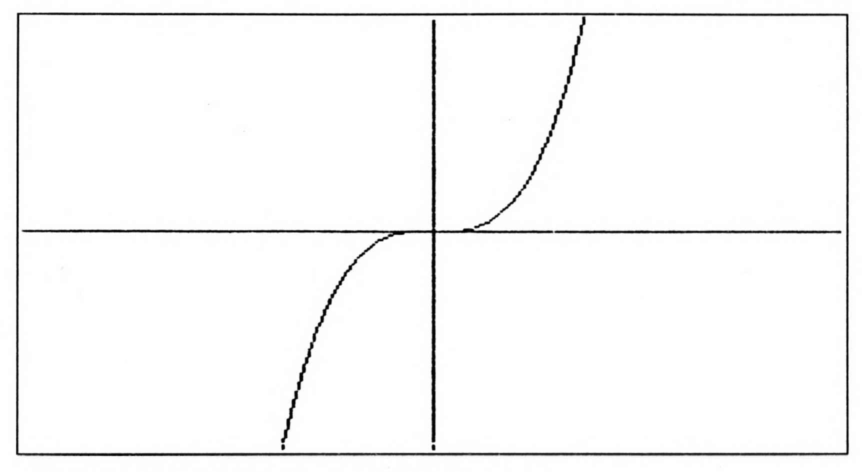


Abb. 5.10: Bildschirmdarstellung der Kubik-Funktion

```
100 REM ------------------------------
110 REM ---   Der Weg zum Kreis     ---
120 REM ------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 100,200
220 x=100
230 :
240 FOR i=0 TO PI STEP 0.008
250    PLOT x,200+180*SIN(i),1
260    PLOT x,200-180*SIN(i),1
270    x=x+1
280 NEXT
```

Programm 5.11: Lösungsansatz zum Zeichnen eines Kreises

Kommen wir nun wieder zu den Winkelfunktionen zurück, und versuchen wir, einen Kreis darzustellen. Einen möglichen Lösungsansatz zeigt Programm 5.11.

Wir nehmen die obere Hälfte der Sinus-Kurve und setzen sie in Bildschirmmitte oberhalb der Mittelachse. Da wir hier den PLOT-Befehl verwenden, muß die Schrittweite noch kleiner als bisher gewählt werden. Wenn wir nun noch den negativen Wert der Sinus-Funktion zusätzlich unterhalb der Mittelachse

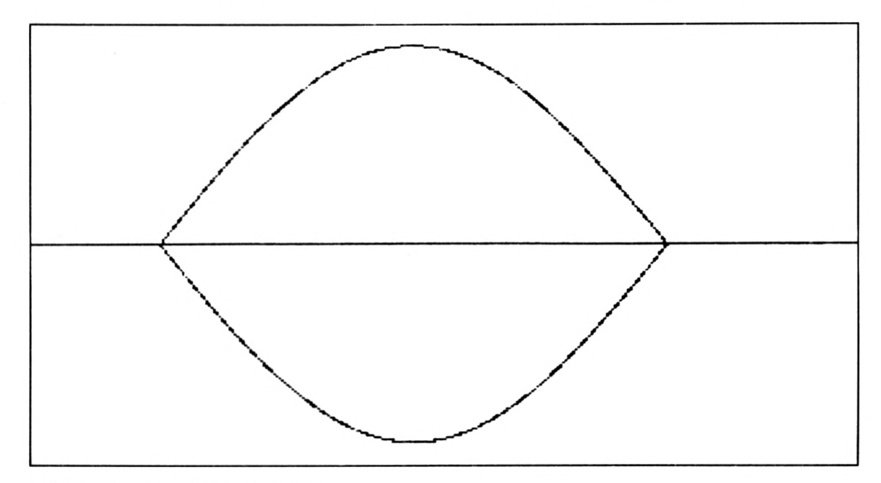


Abb. 5.11: Angenäherter Kreis

```
100 REM ------------------------------
110 REM ---  Der Weg zum Kreis      ---
120 REM ------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 MOVE 0,200
200 DRAW 639,200,1
210 MOVE 100,200
220 x=100
230 :
240 FOR i=0 TO 2*PI STEP 0.008
250    PLOT 320+180*COS(i),200+180*SIN(i),1
280 NEXT
```

Programm 5.12: Kreis-Programm

ausgeben, so müßte doch eigentlich ein Kreis entstehen, oder? Sehen Sie selbst (Abb. 5.11).

Wer sich die Sinus-Funktion genau anschaut, wird feststellen, daß sie nicht aus Halbkreisen besteht, sondern eher aus dem Teil einer Ellipse. Daher ergeben sich in Abb. 5.11 auch die „Ecken“ an der rechten und linken Seite. Wie kann man dem nun abhelfen?

Wie wir gesehen haben, verläuft die Cosinus-Funktion etwas versetzt zur Sinus-Funktion. Außerdem haben wir bisher die anzusprechenden Punkte auf

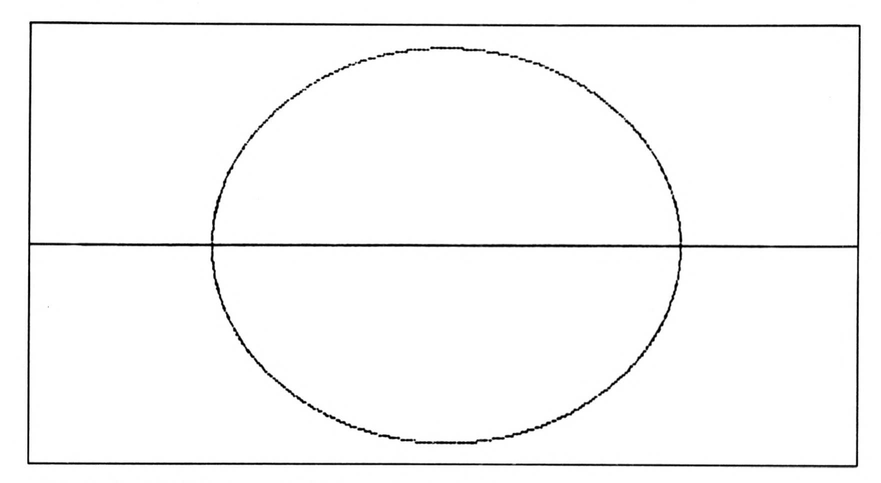


Abb. 5.12: Mit Programm 5.12 gezeichneter Kreis

der X-Achse immer fortgeschrieben (x=x+1). Versuchen wir es doch einmal, indem wir die Werte der X-Achse mit der Cosinus-Funktion belegen (Programm 5.12).

Wie uns Abb. 5.12 zeigt, haben wir unser Ziel erreicht. Betrachten Sie beim Zeichnen am Bildschirm genau die Reihenfolge des Setzens der Punkte, sie geben Ihnen einen Aufschluß über den Vorgang. Probieren Sie auch die Kombinationen COS/COS, SIN/SIN sowie SIN/COS. Welche Kombinationen führen noch zum Ziel?

Nun können wir auch die Mittellinie weglassen, wie das nächste Listing und die zugehörige Abbildung (Abb. 5.13) zeigen.

```
100 REM -------------------------------
110 REM ---          Kreis          ---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 :
200 FOR i=0 TO 2*PI STEP 2*PI/60
210    IF i=0 THEN MOVE 500,200
220    DRAW 320+180*COS(i),200+180*SIN(i),1
230 NEXT
```

Programm 5.13: Zeichnen eines Kreises (ohne Mittellinie)

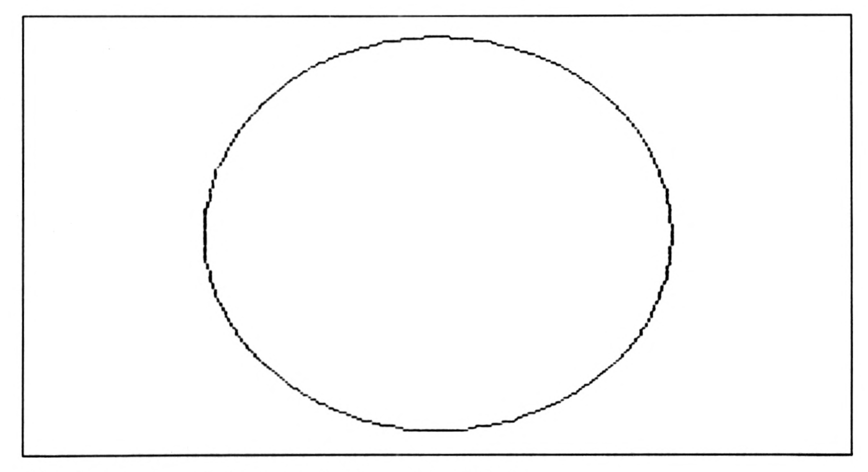


Abb. 5.13: Der mit Programm 5.13 gezeichnete Kreis

Eigentlich läßt sich kein Unterschied zwischen den Abbildungen 5.12 und 5.13 feststellen, obwohl wir noch eine weitere Programmänderung vorgenommen haben. Beim Zeichnen am Bildschirm ist Ihnen sicherlich aufgefallen, daß die neue Version wesentlich schneller erstellt ist. Hier haben wir wieder den DRAW-Befehl verwendet. Damit Sie jetzt zu Beginn keinen störenden Strich auf dem Bildschirm finden, haben wir in Zeile 210 noch den Grafik-Cursor auf den ersten Punkt des Kreises gesetzt. Dies läßt sich leicht nachrechnen, da beim Wert 0 die Cosinus-Funktion den Wert 1 hat und somit 500=320+180 ergibt. Der Sinus hat beim Argument 0 ebenfalls den Wert 0, so daß hier nur die Konstante 200 zählt.

Auch die Schrittweite wurde vergrößert: Es werden nur 60 Teillinien zum Zeichnen des gesamten Kreises verwendet.

Im nächsten Listing (Programm 5.14) haben wir nichts – im übertragenen Sinne – weggelassen. Finden Sie es?

Wenn Sie die Änderung anhand des Listings nicht finden, so tippen Sie das Listing exakt so wie abgebildet ein. Sehen Sie die Änderung bei der Bildschirmausgabe? Wenn nicht, schauen Sie sich Abb. 5.14 an.

```
100 REM -------------------------------
110 REM ---         Sechseck        ---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 :
200 FOR i=0 TO 2*PI STEP 2*PI/6
210    IF i=0 THEN MOVE 500,200
220    DRAW 320+180*COS(i),200+180*SIN(i),1
230 NEXT
```

Programm 5.14: Was wurde weggelassen?

Kleine Änderungen, große Wirkung, wie Sie sicherlich schon oft in diesem Zusammenhang festgestellt haben.

Mathematiker wissen, daß ein Kreis der Spezialfall einer anderen geometrischen Figur ist: der Ellipse. Auch Sie werden dies gleich feststellen, wenn Sie Programm 5.15 abtippen.

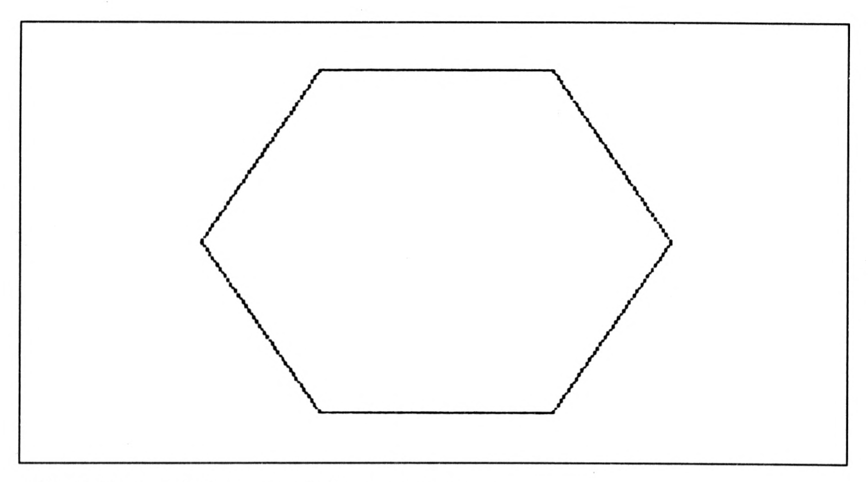


Abb. 5.14: Durch Programm 5.14 erzeugte Figur

Ist Ihnen der Unterschied aufgefallen? Bisher haben wir die Funktionswerte von Cosinus und Sinus jeweils mit dem gleichen Wert multipliziert, im letzten Listing haben wir dies anders gemacht (320 für Cosinus und 50 für Sinus). Das Ergebnis ist ein in der Höhe zusammengestauchter Kreis – eine Ellipse – wie die Abb. 5.15 zeigt.

Mit diesen Werten wollen wir nun etwas herumspielen. Wir werden im nächsten Beispielprogramm zwei Ellipsen benutzen, wobei wir die eine zum Setzen des Grafik-Cursors verwenden und zum aktuellen Wert der anderen Ellipse eine Linie ziehen (Programm 5.16).

```
100 REM ------------------------------
110 REM ---        Ellipse         ---
120 REM ------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 :
200 FOR i=0 TO 2*PI STEP 2*PI/60
210    IF i=0 THEN MOVE 640,200
220    DRAW 320+320*COS(i),200+50*SIN(i),1
230 NEXT
```

Programm 5.15: Zeichnen einer Ellipse

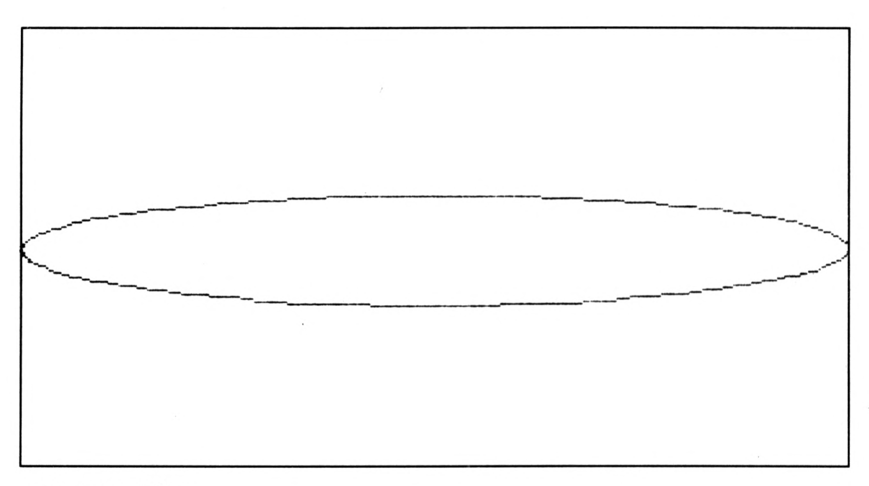


Abb. 5.15: Ellipse

Das Ergebnis zeigt Abb. 5.16, wobei wir wegen der Übersichtlichkeit die beiden imaginär verwendeten Ellipsen ebenfalls dargestellt haben. Betrachten

Sie sich das Listing und die Abbildung genau, und versuchen Sie herauszufinden, in welcher Reihenfolge die beiden Ellipsen durchlaufen werden. Achtung, wir haben in Zeile 210 negative Vorzeichen verwendet.

Sollte Ihnen dies nicht sofort klar werden, so können Sie auch das letzte Listing verwenden und die Parameter entsprechend ändern.

```
100 REM ------------------------------
110 REM --- Linien zwischen Ellipsen---
120 REM ------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 :
200 FOR i=0 TO 2*PI STEP 2*PI/60
210    MOVE 320+320*COS(i),200+50*SIN(i)
220    DRAW 320+50*COS(i),200+200*SIN(i),1
230 NEXT
```

Programm 5.16: Linien zwischen Ellipsen (Version 1)

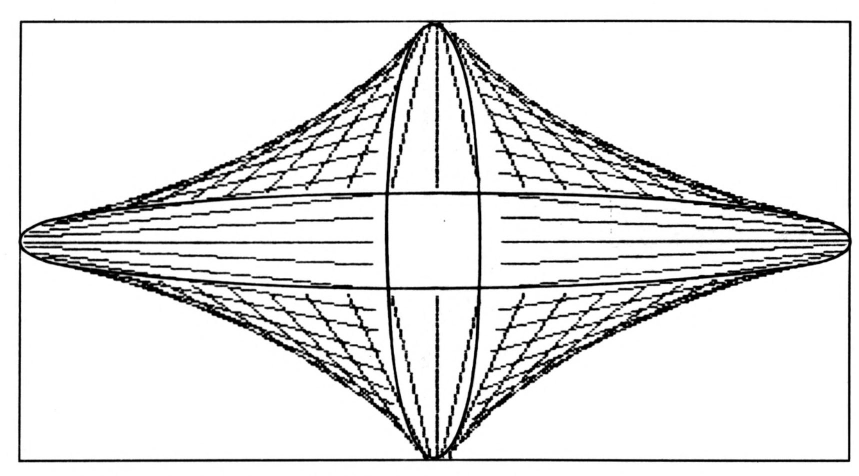


Abb. 5.16: Die von Programm 5.16 erzeugte Grafik

Was passiert nun, wenn wir bei einer Ellipse die Werte von einer Koordinate quadrieren (Programm 5.17)?

Wie wir bereits wissen, gibt es in diesem Fall keine negativen Werte. Die Nachverfolgung der einzelnen Linien wird in Abb. 5.17 schon etwas schwieriger. Mit unseren kurzen Programmbeispielen haben wir also in den letzten beiden Fällen schon hübsche Grafiken gezeichnet.

Im nächsten Beispiel haben wir noch höhere Potenzen verwendet, aber sehen Sie selbst (Programm 5.18).

```
100 REM -------------------------------
110 REM --- Linien zwischen Ellipsen---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 :
200 FOR i=0 TO 2*PI STEP 2*PI/60
210    MOVE 320-320*COS(i)^2,200-50*SIN(i)
220    DRAW 320+50*COS(i),200+200*SIN(i),1
230 NEXT
```

Programm 5.17: Linien zwischen Ellipsen (Version 2)

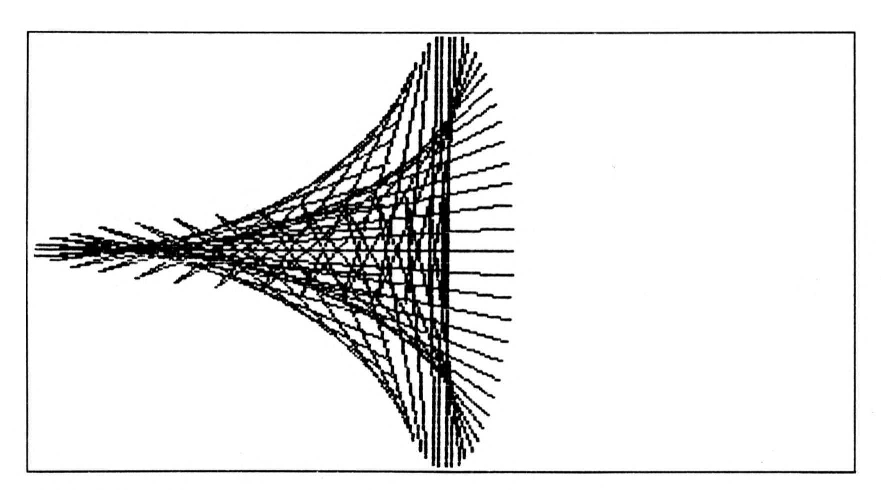


Abb. 5.17: Die von Programm 5.17 erzeugte Grafik

Bevor wir nun zu einer anderen Variante des Kreises kommen, noch ein kleines zweifarbiges Beispiel. Einige der zuletzt beschriebenen Bilder werden wir im weiteren nochmals in ausgebauter Form kennenlernen. Da wir hier jedoch nur eine grundlegende Einführung geben wollten, sei das Thema Ellipse und Kreis mit den nächsten beiden Beispielen abgeschlossen (Programme 5.19 und 5.20).

Hier noch ein sehr komplexes Beispiel für ineinandergeschachtelte Kreise (Programm 5.20).

```
100 REM -------------------------------
110 REM --- Linien zwischen Ellipsen---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 :
200 FOR i=0 TO 2*PI STEP 2*PI/60
210    MOVE 320-320*COS(i)^15,200-50*SIN(i)
220    DRAW 320+50*COS(i)^7,200+200*SIN(i),1
230 NEXT
```

Programm 5.18: Linien zwischen Ellipsen (Version 3)

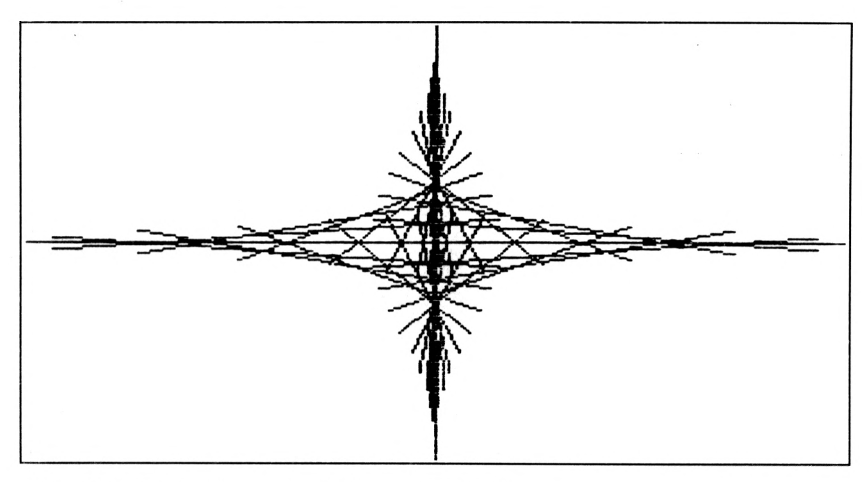


Abb. 5.18: Die von Programm 5.18 erzeugte Grafik

```
100 REM -------------------------------
110 REM --- Linien zwischen Ellipsen---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,15
170 INK 0,0
180 :
190 :
200 FOR i=0 TO 2*PI STEP 2*PI/60
210    MOVE 320+300*SIN(i),200+50*COS(i)
220    DRAW 320+50*SIN(i),200+180*COS(i),1
225    DRAW 320-50*SIN(i),200-180*COS(i),2
230 NEXT
```

Programm 5.19: Linien zwischen Ellipsen (Version 4)

Wie aus Abb. 5.20 ersichtlich ist, wird zunächst ein Kreis um den Bildschirmmittelpunkt gezogen. Dann werden auf dem Umfang dieses Kreises sechs weitere Kreise gezeichnet und auf diesen Kreisumfängen jeweils weitere Kreise. Das Ganze wiederholt sich noch einmal.

Sofern Sie sich mit Kapitel 2 noch nicht beschäftigt haben, empfehlen wir Ihnen hier die Lektüre dieses Kapitels, da das Unterprogramm in seiner allge-

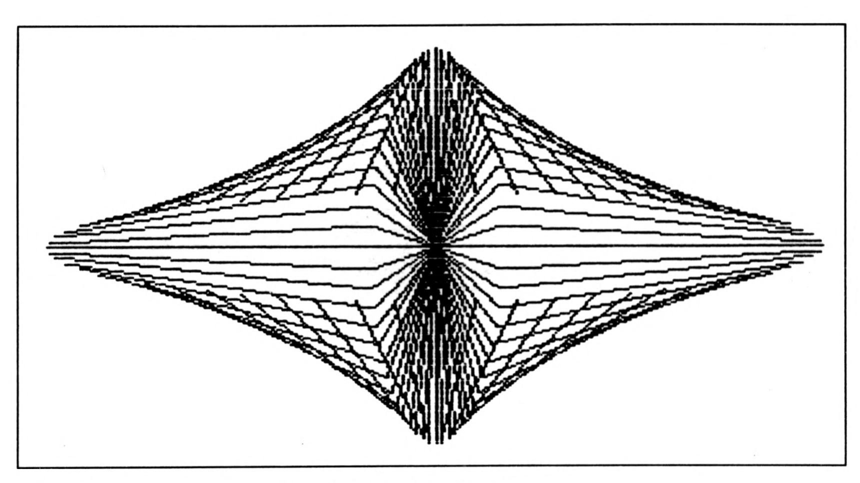


Abb. 5.19: Die von Programm 5.19 erzeugte Grafik

```
100 REM -------------------------------
110 REM --- Vierfach geschachtelte  ---
120 REM ---          Kreise         ---
130 REM -------------------------------
140 :
150 DEG
160 :
170 mittelx = 320
180 mittely = 200
190 radiusx = 149
200 radiusy = 149
210 kanf    = 0
220 kend    = 360
230 kstep   = 10
240 kfarbe  = 1
250 :
260 GOSUB 1000
270 :
280 kfarbe  = 2
290 :
300 FOR i=90 TO 449 STEP 60
310    mittelx = 320+149*SIN(i)
320    mittely = 200+149*COS(i)
330    radiusx = 50
340    radiusy = 50
350    kstep   = 15
360    GOSUB 1000
370 :
380    kfarbe = 3
```

Programm 5.20: Geschachtelte Kreise

```
390 :
400      FOR j=0 TO 359 STEP 60
410         kstep   = 20
420         mittelx = 320+149*SIN(i)+50*SIN(j)
430         mittely = 200+149*COS(i)+50*COS(j)
440         radiusx = 25
450         radiusy = 25
460         GOSUB 1000
470 :
480         kfarbe = 1
490 :
500         FOR k=0 TO 359 STEP 60
510            kstep   =  25
520            mittelx = 320+149*SIN(i)+50*SIN(j)+25*SIN(
k)
530            mittely = 200+149*COS(i)+50*COS(j)+25*COS(
k)
540            radiusx = 12.5
550            radiusy = 12.5
560            GOSUB 1000
570 :
580            kfarbe = 2
590 GOTO 700
600 :
610            FOR m=0 TO 360 STEP n
620               kstep   = 30
630               mittelx = 320+239*SIN(i)+80*SIN(j)+40*S
IN(k)+20*SIN(m)
640               mittely = 200+149*COS(i)+50*COS(j)+25*C
OS(k)+12.5*COS(m)
650               radiusx = 10
660               radiusy = 6.25
670               GOSUB 1000
680            NEXT
690 :
700            kfarbe = 1
710 :
720         NEXT
730 :
740         kfarbe = 3
750 :
760      NEXT
770 :
780      kfarbe = 2
790 :
800 NEXT
810 :
820 END
830 :
1000 REM ----------------------------
1010 REM ---   Kreis im Bogenmass   ---
1020 REM ----------------------------
1030 :
1040     FOR klauf=kanf TO kend STEP kstep
1050        punktx=mittelx+radiusx*SIN(klauf)
1060        punkty=mittely+radiusy*COS(klauf)
```

Programm 5.20: Geschachtelte Kreise (Forts.)

```
1070       IF klauf=kanf THEN MOVE punktx,punkty : GOTO
 1090
1080       DRAW punktx,punkty,kfarbe
1090     NEXT
1100 :
1110 RETURN
```

Programm 5.20: Geschachtelte Kreise (Forts.)

meinen Form in diesem Kapitel erläutert ist. Auch steht dieses Beispiel stellvertretend für Angaben wie Gradmaß, wie Sie aus Zeile 150 ersehen.

In den Zeilen ab 170 werden die Variablen für das Unterprogramm ab Zeile 1000 besetzt. Man muß also nicht immer direkt beim eigentlichen Zeichnen der Grafiken die Variablen angeben, sondern man kann sie auch gesammelt am Anfang des Programms definieren. Wer viel mit den Grafiken herumprobiert, sollte das Einlesen der Daten auch mittels des INPUT-Befehls ausführen.

Nachdem der größte Kreis gezeichnet wurde (Aufruf in Zeile 260), wird zunächst die Farbe umgeschaltet, um dann – beginnend an der rechten Seite des großen Kreises – die sechs weiteren Kreise zu zeichnen. Dafür werden in der FOR...NEXT-Schleife ab Zeile 300 die Variablen zum Aufruf des Unterprogramms entsprechend vorbesetzt. Hier wird auch der Sinn des Unterpro-

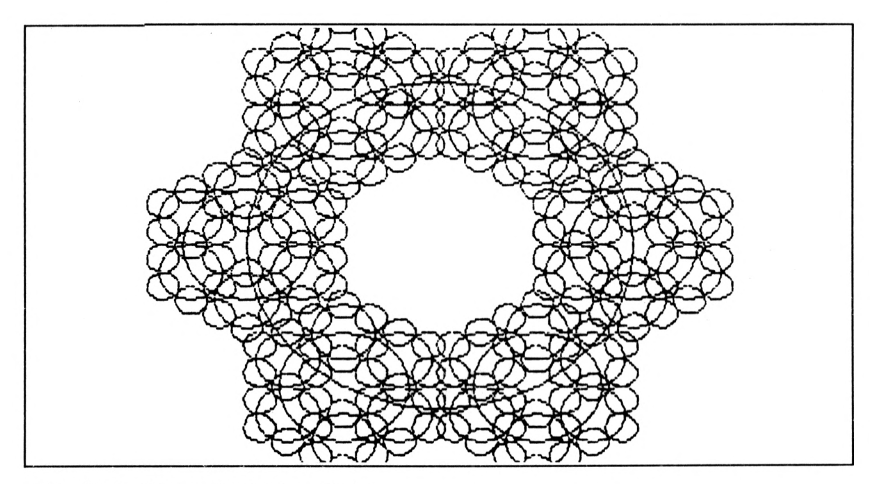


Abb. 5.20: Die geschachtelten Kreise

gramms deutlich: Man braucht nicht immer das Zeichnen eines Kreises explizit durchzuführen, es genügt das Festlegen der Parameter und anschließend der Unterprogrammaufruf.

In gleicher Weise wie ab Zeile 280 wird ab Zeile 380 vorgegangen. Hier wird die Farbe zunächst auf 3 geschaltet und in der FOR...NEXT-Schleife ab Zeile 400 – diesmal wieder im Kreis oben beginnend – der gesamte Parametersatz den neuen Gegebenheiten angepaßt. Die Programmsequenz wiederholt sich – mit anderen Parametern – ab Zeile 480, wo die letzte Gruppe der Kreise gezeichnet wird. In unserem Beispielprogramm verlassen wir in Zeile 590 den normalen Programmablauf und springen gleich zur Zeile 700, wo die einzelnen Schleifen beendet werden.

Natürlich muß auch jeweils die Farbe umgesetzt werden, um allen Kreisen der gleichen Größe auch die gleiche Farbe zuzuordnen. Wer will, kann noch eine weitere Schleife einbauen, wie sie ab Zeile 600 vorgegeben ist. Die Grafik wird allerdings dann recht unübersichtlich. Probieren Sie auch verschiedene zwischen den Kreisen liegende Winkel aus.

Auch das Unterprogramm zum Zeichnen der Sterne aus Kapitel 2 wollen wir hier modifiziert wiedergeben (Programm 5.21). Allerdings haben wir hier das Unterprogramm nicht auf Basis der Parameter für das Zeichnen eines Kreises aufgebaut, sondern eine kürzere Form gewählt. Zur näheren Erklärung des Unterprogramms aus Zeile 1000 lesen Sie bitte in Kapitel 2 nach.

```
100 REM -------------------------------
110 REM --- Dreifach gestaffelte    ---
120 REM ---             Sterne      ---
130 REM -------------------------------
140 :
150 MODE 1
160 :
170 INK 1,6
180 INK 2,15
190 INK 3,18
200 :
210 farbe=1
220 :
230 x=320
240 y=200
250 a=0
260 :
270 abstand=80
280 GOSUB 1000
```

Programm 5.21: Dreifach gestaffelte Sterne

```
290 :
300 farbe=2
310 :
320 FOR j=0 TO 2*PI STEP PI/3
330    x=320+240*SIN(j)
340    y=200+120*COS(j)
350    a=0
360    abstand=40
370    GOSUB 1000
380 :
390    farbe=3
400 :
410    FOR k=0 TO 2*PI STEP PI/3
420       x=320+240*SIN(j)+120*SIN(k)
430       y=200+120*COS(j)+60*COS(k)
440       a=0
450       abstand=20
460       GOSUB 1000
470    NEXT
480 :
490    farbe = 2
500 :
510 NEXT
520 :
530 END
540 :
1000 REM ------------------------------
1010 REM --- Unterprogramm: Stern   ---
1020 REM ------------------------------
1030 :
1040    FOR i=0 TO 2*PI STEP PI/6
1050       IF a=1 THEN abstand=abstand/2 : a=0 : GOTO 1070
1060       IF A=0 THEN abstand=abstand*2 : a=1
1070       kx=X+abstand*SIN(i)
1080       ky=Y+abstand/2*COS(i)
1090       IF I=0 THEN MOVE KX,KY : GOTO 1110
1100       DRAW kx,ky,farbe
1110    NEXT
1120 :
1130 RETURN
```

Programm 5.21: Dreifach gestaffelte Sterne (Forts.)

Zunächst wird auch hier wieder der 4-Farben-Modus eingeschaltet. Auch wenn Sie diesen bereits eingeschaltet haben, sollten Sie den Befehl verwenden, da dadurch auch der Bildschirm gelöscht wird. Dann werden die Farben neu gewählt und einige Parameter vorbesetzt. Beachten Sie, daß der Wert der Variablen abstand im Unterprogramm ab Zeile 1000 sowohl verdoppelt als auch halbiert wird. Dies ist unbedingt zu berücksichtigen, wenn die Größe des Sterns vorgegeben sein soll.

Gegenüber den vierfach gestaffelten Kreisen im letzten Beispiel haben wir hier nur eine dreifache Staffelung der Sterne durchgeführt, um das Bild nicht allzu chaotisch werden zu lassen (siehe Abb. 5.21).

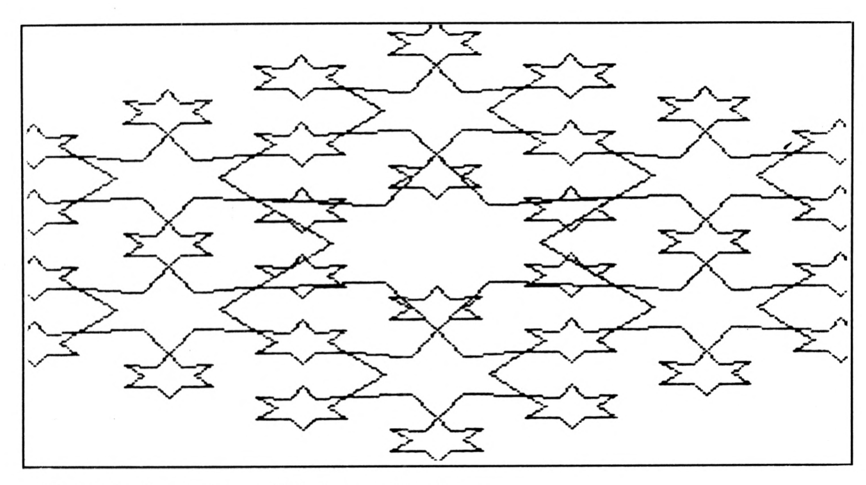


Abb. 5.21: Dreifach gestaffelte Sterne, Hardcopy

Dementsprechend befinden sich in diesem Programm auch nur zwei FOR-...NEXT-Schleifen, da der innere Stern keiner Schleife bedarf. Achten Sie auf die Schrittweite der FOR...NEXT-Schleifen. Sie sind dafür maßgebend, daß sich die kleineren Sterne jeweils an einem Zacken des größeren Sterns befinden. Durch die Schrittweite in Zeile 1040 können Sie die Anzahl der Zacken im Stern bestimmen, wie wir im nächsten Beispiel sehen (Programm 5.22).

```
100 REM -------------------------------
110 REM --- Doppelt  gestaffelte    ---
120 REM ---             Sterne      ---
130 REM -------------------------------
140 :
150 MODE 1
160 :
170 INK 1,6
180 INK 2,15
190 :
200 farbe=1
210 :
220 x=320
230 y=200
240 a=0
250 :
260 abstand=80
270 GOSUB 1000
280 :
290 farbe=2
300 :
```

Programm 5.22: Doppelt gestaffelte Sterne

```
310 FOR j=0 TO 2*PI STEP PI/6
320     x=320+240*SIN(j)
330     y=200+120*COS(j)
340     a=0
350     abstand=40
360     GOSUB 1000
370 :
380 NEXT
390 :
400 END
410 :
1000 REM ------------------------------
1010 REM --- Unterprogramm: Stern    ---
1020 REM ------------------------------
1030 :
1040     FOR i=0 TO 2*PI STEP PI/12
1050        IF a=1 THEN abstand=abstand/2 : a=0 : GOTO 1
070
1060        IF A=0 THEN abstand=abstand*2 : a=1
1070        kx=X+abstand*SIN(i)
1080        ky=Y+abstand/2*COS(i)
1090        IF I=0 THEN MOVE KX,KY : GOTO 1110
1100        DRAW kx,ky,farbe
1110     NEXT
1120 :
1130 RETURN
```

Programm 5.22: Doppelt gestaffelte Sterne (Forts.)

Die hierzu korrespondierende Abbildung ist Abb. 5.22.

Hier haben wir eine weitere Schleife gestrichen, so daß nur noch eine FOR-...NEXT-Schleife vorhanden ist. Gegenüber dem vorigen Listing wurden nur

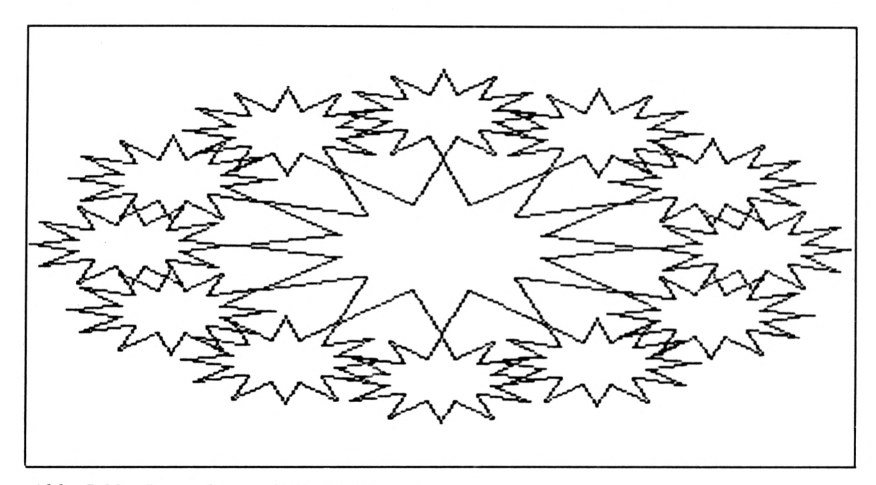


Abb. 5.22: Doppelt gestaffelte Sterne, Hardcopy

die Schrittweiten und Abstände geändert. Einen zwölfzackigen Stern erhalten Sie also mit der Schrittweite PI/12 in Zeile 1040. Sollen die umgebenden Sterne jeweils an einem Zacken den größeren Stern berühren, so ist die Schrittweite hierfür die Hälfte der Schrittweite im Unterprogramm ab Zeile 2000.

Im nächsten Beispiel wollen wir noch einen Schritt in extremer Richtung weiter gehen und 120zackige Sterne darstellen. Da hier das in den letzten beiden Listings verwendete Verfahren zu keiner schönen Grafik führt, wollen wir drei solcher Sterne ineinanderlagern, wie Abb. 5.23 zeigt.

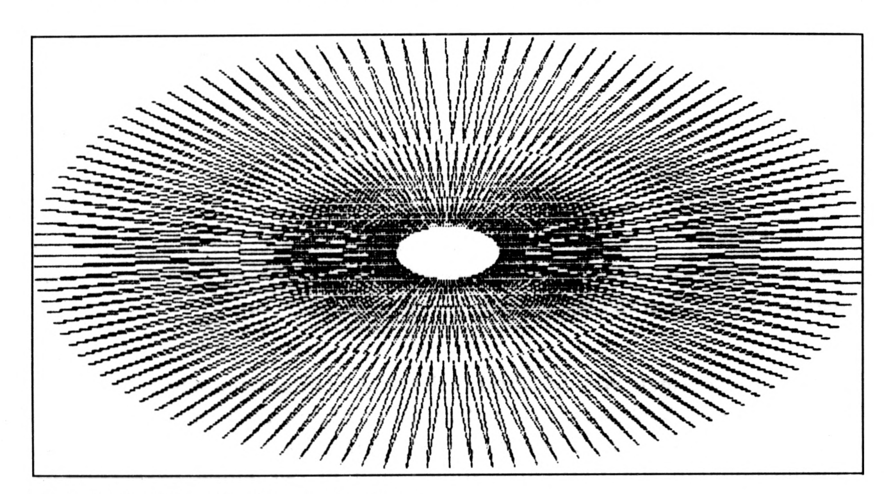


Abb. 5.23: 120zackige Sterne, Hardcopy

```
100 REM -------------------------------
110 REM --- 120-zackige Sterne      ---
120 REM -------------------------------
130 :
140 MODE 1
150 :
160 INK 1,6
170 :
180 farbe=1
190 :
200 x=320
210 y=200
220 a=0
230 :
240 abstand=160
250 GOSUB 1000
260 :
```

Programm 5.23: 120zackige Sterne

```
270 abstand=80
280 GOSUB 1000
290 :
300 abstand=40
310 GOSUB 1000
320 :
330 END
340 :
1000 REM ------------------------------
1010 REM --- Unterprogramm: Stern   ---
1020 REM ------------------------------
1030 :
1040    FOR i=0 TO 2*PI STEP PI/120
1050       IF a=1 THEN abstand=abstand/2 : a=0 : GOTO 1
070
1060       IF A=0 THEN abstand=abstand*2 : a=1
1070       kx=X+abstand*SIN(i)
1080       ky=Y+abstand/1.6*COS(i)
1090       IF I=0 THEN MOVE KX,KY : GOTO 1110
1100       DRAW kx,ky,farbe
1110    NEXT
1120 :
1130 RETURN
```

Programm 5.23: 120zackige Sterne (Forts.)

Bei Ihrem bisherigen Wissensstand bedarf das Listing wohl keiner weiteren Erklärung mehr. Vielleicht beschreiben wir noch den Trick mit der Ausdehnung der Sterne bis an den Bildschirmrand. Da in der Breite (X-Richtung) 640 Bildschirmpunkte zur Verfügung stehen und in der Höhe (Y-Richtung) 400 Bildschirmpunkte, muß natürlich in Zeile 1080 der Cosinus auch mit dem entsprechenden Faktor versehen werden. Dieser ist 1,6 (640/400).

```
100 REM -------------------------------
110 REM --- Sterne mit unterschied- ---
120 REM --- lichen Zacken           ---
130 REM -------------------------------
140 :
150 MODE 1
160 :
170 INK 1,6
180 :
190 FARBE=1
200 :
210 X=320
220 Y=200
230 :
240 FOR J=50 TO 250 STEP 20
```

Programm 5.24: Sterne mit verschieden langen Zacken

```
250    A=0
260    ABSTAND = J
270    GOSUB 1000
280 NEXT
290 :
300 END
310 :
1000 REM -----------------------------
1010 REM --- Stern mit verschieden  ---
1020 REM ---        langen Zacken   ---
1030 REM -----------------------------
1040 :
1050     abstand0 = abstand
1060     abstand1 = abstand*3/4
1070     abstand2 = abstand/2
1080 :
1090     FOR i=0 TO 2*PI STEP 2*PI/32
1100        IF a=0 THEN abstand=abstand2 : a=1 : GOTO 11
40
1110        IF A=1 THEN abstand=abstand0 : a=2 : GOTO 11
40
1120        IF A=2 THEN abstand=abstand2 : a=3 : GOTO 11
40
1130        IF A=3 THEN abstand=abstand1 : a=0
1140        kx=X+abstand*SIN(i)
1150        ky=Y+abstand*COS(i)
1160        IF I=0 THEN MOVE KX,KY : GOTO 1180
1170        DRAW kx,ky,farbe
1180     NEXT
1190 :
1200 RETURN
```

Programm 5.24: Sterne mit verschieden langen Zacken (Forts.)

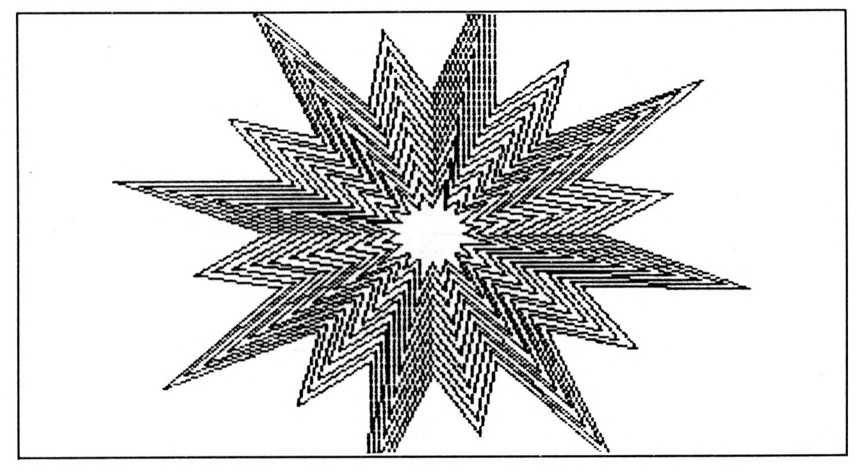


Abb. 5.24: Sterne mit verschieden langen Zacken, Hardcopy

Im vorletzten Beispiel dieses Kapitels (Programm 5.24) wollen wir die Form des Sterns noch einmal verändern, so daß zwei verschieden lange Zackenformen entstehen. Dazu brauchen wir natürlich drei verschieden große imaginäre Kreise, die durch die drei Abstände in den Variablen abstand0, abstand1 und abstand2 berechnet werden. Auch hier sollten Sie wieder die Parameter ändern, in diesem Fall unter anderem die Faktoren und Dividenden bei den Abständen in den Zeilen 1060 und 1070.

Natürlich ist auch der Aufwand zur Bestimmung des aktuellen Abstands etwas größer (ab Zeile 1100).

Das Ergebnis der Testdaten ab Zeile 240 finden Sie in Abb. 5.24.

Abschließend in diesem Einführungskapitel wollen wir nun einen Stern mit zwei verschiedenen Zackenlängen und zwei verschiedenen Farben derart darstellen, daß eine Schattierung entsteht (Programm 5.25).

```
100 REM -------------------------------
110 REM --- Sterne mit unterschied- ---
120 REM --- lichen Zacken und Farben---
130 REM -------------------------------
140 :
150 MODE 1
160 :
170 INK 0,0
180 INK 1,1
190 INK 2,2
200 :
210 FARBE=1
220 :
230 X=320
240 Y=200
250 :
260 FOR J=0 TO 200
270   A=1
280   ABSTAND = J
290   GOSUB 1000
300 NEXT
310 :
320 END
330 :
1000 REM -----------------------------
1010 REM --- Stern mit verschieden  ---
1020 REM ---         langen Zacken   ---
1030 REM -----------------------------
1040 :
1050    abstand0 = abstand
1060    abstand1 = abstand*3/4
1070    abstand2 = abstand/2
1080 :
```

Programm 5.25: Sterne mit zwei verschiedenen Zackenlängen und zwei verschiedenen Farben

```
1090     FOR i=0 TO 2*PI STEP 2*PI/16
1100        IF a=0 THEN abstand=abstand2 : a=1 : GOTO 11
40
1110        IF A=1 THEN abstand=abstand0 : a=2 : GOTO 11
40
1120        IF A=2 THEN abstand=abstand2 : a=3 : GOTO 11
40
1130        IF A=3 THEN abstand=abstand1 : a=0
1140        kx=X+abstand*SIN(i)
1150        ky=Y+abstand*COS(i)
1160        IF I=0 THEN MOVE KX,KY : GOTO 1200
1170        IF farbe=1 THEN farbe=2 : GOTO 1190
1180        IF farbe=2 THEN farbe=1
1190        DRAW kx,ky,farbe
1200     NEXT
1210 :
1220 RETURN
```

Programm 5.25: Sterne mit zwei verschiedenen Zackenlängen und zwei verschiedenen Farben (Forts.)

Abb. 5.25 gibt hier nur eine ungenügende Darstellung, da Hardcopys leider nicht in Farbe wiedergegeben werden können.

Erreicht wird das Ganze zunächst durch die geschickte Farbwahl in den Zeilen 180 und 190 (blau/hellblau). Dann wird in den Zeilen 1170 und 1180 jeweils die Farbe umgeschaltet, so daß die Farben abwechselnd zum Tragen kommen.

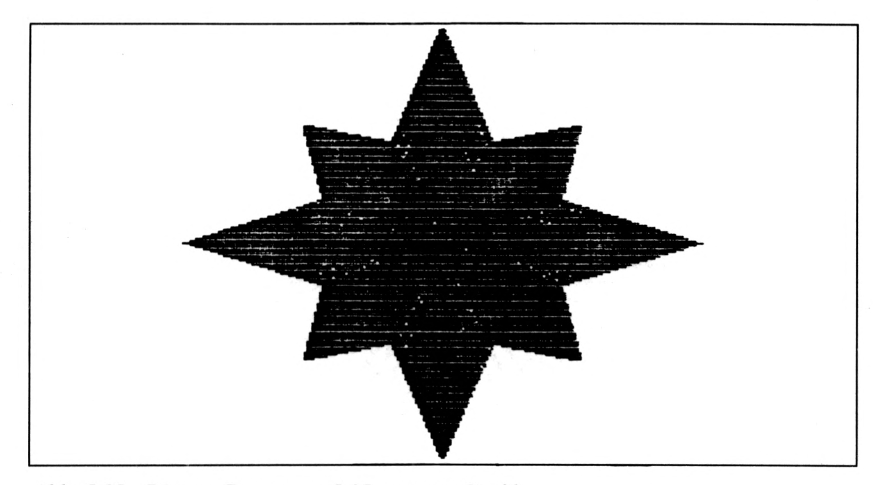


Abb. 5.25: Die von Programm 5.25 erzeugte Grafik

5.2 LAUFENDE FARBEN

Im vorliegenden Kapitel wollen wir uns mit einer Eigenschaft der CPC-Rechner beschäftigen, die sich besonders im Modus 0 auswirkt, da hier sechzehn Farben vorliegen. In einigen Beispielen wollen wir Effekte vorführen, die durch periodisches Durchlaufen der Farben entstehen. Als einfaches Anfangsbeispiel wollen wir einen Kreisring aus lauter Linien (Radien) wählen (Programm 5.26).

Zunächst wird in Zeile 150 auf den 16-Farben-Modus umgeschaltet. Dann bekommen alle zulässigen INK-Nummern die Farbe entsprechend ihrer Nummer zugeordnet. Damit Sie etwas das Aussehen des Kreisringes variieren können, werden die Variablen x und y in den Zeilen 210 und 220 vorbesetzt.

Aus Kapitel 2 dürfte Ihnen das Zeichnen eines Kreises bekannt sein. Zunächst

```
100 REM -------------------------------
110 REM --- Laufende Farben am      ---
120 REM ---               Kreisring ---
130 REM -------------------------------
140 :
150 MODE 0
160 :
170 FOR i=0 TO 15
180    INK i,i
190 NEXT
200 :
210 x = 200
220 y = 100
230 :
240 REM -------------------------------
250 REM ---   Zeichnen der Figur    ---
260 REM -------------------------------
270 :
280 FOR i=0 TO 2*PI STEP 0.1
290   MOVE 320+x*SIN(i),200+x*COS(i)
300   f=f+1
310   IF f=16 THEN f=1
320   DRAW 320+y*SIN(i),200+y*COS(i),f
330 NEXT
340 :
350 REM -------------------------------
360 REM --- 'Laufen' der Farben     ---
370 REM -------------------------------
380 :
390 FOR j=1 TO 100
400    FOR i=1 TO 15
410    INK i,(i+j) MOD 15
420    NEXT
430 NEXT
440 CALL &A080
```

Programm 5.26: Laufende Farben in einer Kreisfigur

wird durch den MOVE-Befehl in Zeile 290 der Grafik-Cursor jeweils an den Anfang der Linie am Rand des inneren Kreises gesetzt. Dann wird die Farbnummer erhöht und gegebenenfalls zurückgeschaltet (Zeile 310). Anschließend wird die Linie zum gleichen Winkel des äußeren Kreises gezogen.

Der wichtigste Teil in dem Beispiel befindet sich ab Zeile 390, da hier 100mal jeweils die Farben fortgeschaltet werden. Den Abschluß in Zeile 440 bildet der Aufruf eines Hardcopy-Befehls, den wir in den folgenden Beispielen allerdings weglassen wollen.

Sicherlich ist die Hardcopy (Abb. 5.26) kein geeignetes Mittel, um Ihnen die Wirkung der Grafik am Bildschirm zu veranschaulichen. Trotzdem wollen wir Ihnen die Hardcopy-Ausgaben zu den einzelnen Beispielen ausdrucken, um Ihnen eine Kontrolle über das Aussehen der Grafik zu ermöglichen. Selbst eine farbige Darstellung würde hier die Schönheit des Effektes nicht wiedergeben können.

Im folgenden Beispiel (Programm 5.27) wollen wir den Effekt an einer Ellipse aus Radien – diesmal vom Mittelpunkt aus – darstellen. Da jedoch die Farben im Kreisring durch die direkte Umsetzung der Farbnummer nicht sehr glücklich sind, wollen wir ein paar Programmzeilen voranstellen, in denen die Farben gesondert aufbereitet werden, wobei in der Reihenfolge der DATA-Zeile 160 die Farben jeweils von links nach rechts dunkler werden.

Zunächst wird das Feld farbe() dimensioniert, und anschließend werden die entsprechenden Farben ausgewählt, die Sie jedoch auch Ihren persönlichen

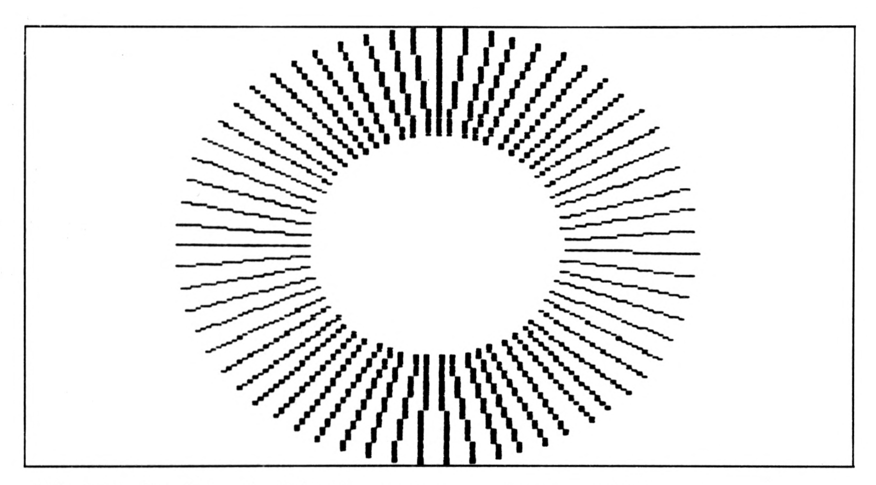


Abb. 5.26: Hardcopy der Bildschirmausgabe von Programm 5.26

```
100 REM ------------------------------
110 REM ---  Farblauf an Ellipse    ---
120 REM ------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
260 INK 0,0
270 :
280 REM ------------------------------
290 REM ---   Zeichnen der Ellipse   ---
300 REM ------------------------------
310 :
320 FOR i=0 TO 2*PI STEP 2*PI/64
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    MOVE 320,200
360    DRAW 320+300*SIN(i),200+150*COS(i),farbe(farbe)
370 NEXT
380 :
390 REM ------------------------------
400 REM --- 'Laufen' der Farben     ---
410 REM ------------------------------
420 :
430 FOR j=1 TO 100
440    FOR i=1 TO 15
450    INK i,farbe(((i+j) MOD 15)+1)
460    NEXT
470 NEXT
```

Programm 5.27: Farblauf an einer Ellipsenfigur

Wünschen anpassen können. Dann werden die Daten eingelesen, die Farbe wird innerhalb des Computers entsprechend umgesetzt und das Feld farbe() besetzt.

Zur Sicherheit wird noch der 16-Farben-Modus eingeschaltet (wenn er es vielleicht vorher nicht war) und – aus optischen Gründen – für einen schwarzen Hintergrund gesorgt (Zeile 260).

Das Zeichnen der Ellipse dürfte für Sie sicherlich kein Problem darstellen. Die Dichte der Linien können Sie durch den Parameter 64 hinter STEP einfach verändern. Das Laufen der Farben wurde entsprechend so umgesetzt, daß nun jeweils die Farben aus dem Feld farbe() im INK-Befehl eingetragen werden.

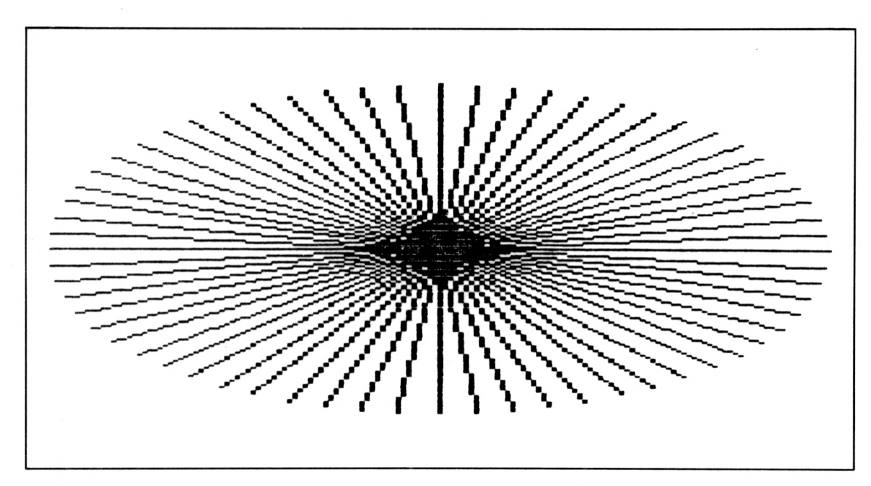


Abb. 5.27: Hardcopy der Bildschirmausgabe von Programm 5.27

Eine weitere Veränderung wäre z. B. das Eintragen von zwei Farben, um noch ein zusätzliches Blinken zu erreichen. Wie bei allen anderen Beispielen auch, sind Ihrer Phantasie keine Grenzen gesetzt.

Den Kontrollausdruck zum Beispiel „Farblauf einer Ellipse“ finden Sie in Abb. 5.27.

```
100 REM -------------------------------
110 REM ---   Farblauf an Figur     ---
120 REM -------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
260 INK 0,0
270 :
280 REM -------------------------------
290 REM ---   Zeichnen der Figur    ---
300 REM -------------------------------
```

Programm 5.28: Farblauf an einer Grafik

```
310 :
320 FOR i=0 TO 2*PI STEP 2*PI/64
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    MOVE 320+ 50*SIN(i),200 + 180*COS(i)
360    DRAW 320+ 300*SIN(i),200 + 50*COS(i),farbe(farbe)
370 NEXT
380 :
390 REM ------------------------------
400 REM --- 'Laufen' der Farben     ---
410 REM ------------------------------
420 :
430 FOR j=1 TO 100
440    FOR i=1 TO 15
450    INK i,farbe(((i+j) MOD 15)+1)
460    NEXT
470 NEXT
```

Programm 5.28: Farblauf an einer Grafik (Forts.)

Ähnlich wie wir beim anfangs vorgestellten Kreisring Linien zwischen zwei verschieden großen Kreisen gezogen haben, wollen wir im nächsten Beispiel dies mit Ellipsen tun (Programm 5.28).

Die einzigen Zeilen, die geändert wurden, sind die Zeilen mit den Nummern 350 und 360. Durch den MOVE-Befehl und den DRAW-Befehl werden zwei verschiedene Ellipsen ausgebildet. Zunächst wird der Grafik-Cursor mit den

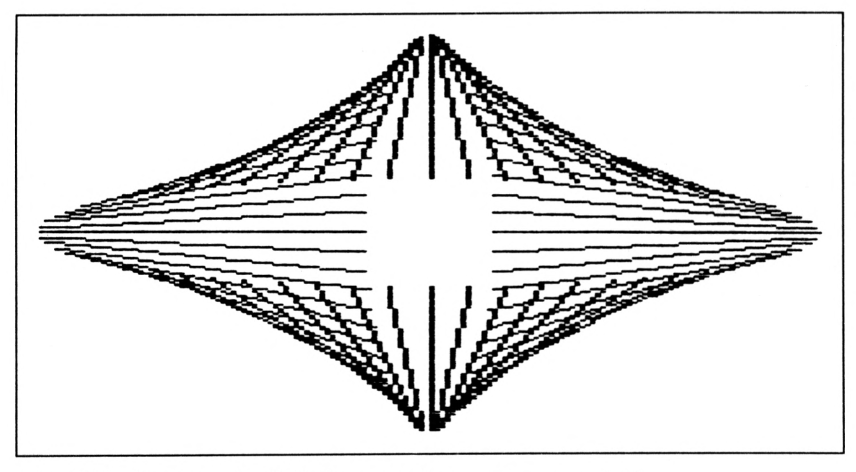


Abb. 5.28: Hardcopy der Bildschirmausgabe von Programm 5.28

Radien 50 (x) und 180 (y) in eine Ellipse um den Mittelpunkt gesetzt. Dann wird eine Linie zu der anderen Ellipse gezogen, die die Radien 300 (x) und 50 (y) hat. Durch die Vertauschung in der Größe der Radien ergeben sich zwei Ellipsen, die senkrecht zueinander stehen, wenn man es vereinfacht ausdrückt.

Die zugehörige Hardcopy-Ausgabe finden Sie in Abb. 5.28. Im nächsten Programm (Programm 5.29) wurde neben den Zeilen 350 und 360 auch noch die Zeile 320 geändert; hier wiederum die Schrittweite zum Abschluß der Zeile (statt 64 jetzt 128). Dadurch, daß im MOVE-Befehl vom Mittelpunkt des Bildschirms aus mit negativen Werten für die Winkelfunktion gearbeitet wurde, ergibt sich ein total anderes Bild, wie Abb. 5.29 zeigt.

```
100 REM -------------------------------
110 REM --- Farblauf an Doppelell.  ---
120 REM -------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
260 INK 0,0
270 :
280 REM -------------------------------
290 REM ---    Zeichnen der Figur    ---
300 REM -------------------------------
310 :
320 FOR i=0 TO 2*PI STEP 2*PI/128
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    MOVE 320-300*SIN(i),200 - 50*COS(i)
360    DRAW 320+ 50*SIN(i),200 + 180*COS(i),farbe(farbe)
370 NEXT
380 :
390 REM -------------------------------
400 REM --- 'Laufen' der Farben     ---
410 REM -------------------------------
420 :
430 FOR j=1 TO 100
440    FOR i=1 TO 15
450    INK i,farbe(((i+j) MOD 15)+1)
460    NEXT
470 NEXT
```

Programm 5.29: Farblauf an einer Doppelellipsen-Grafik

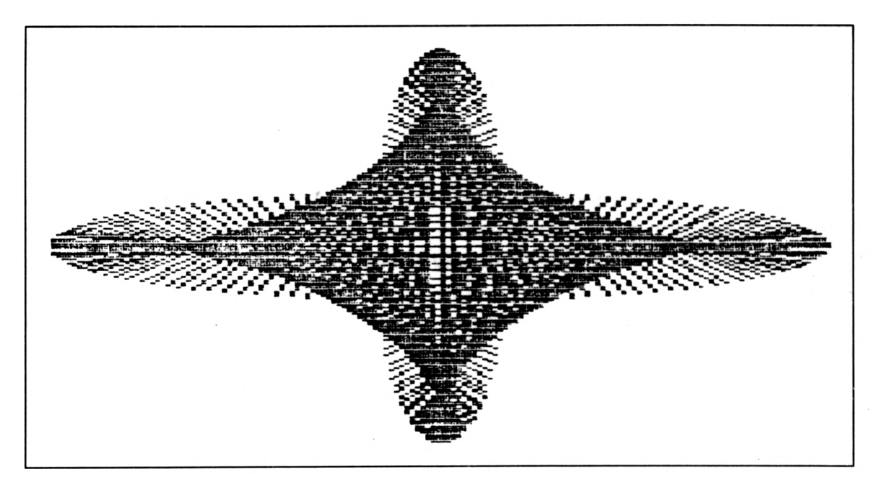


Abb. 5.29: Hardcopy der Bildschirmausgabe von Programm 5.29

Verlassen wir vorerst die Kurvenformen, und wenden wir uns den Rechtecken zu (Programm 5.30). Ein Unterprogramm dazu ist in Kapitel 2 beschrieben, jedoch haben wir hier von der Verwendung der Unterprogramme abgesehen, um Ihnen die Lektüre dieses Kapitels unabhängig von den anderen Kapiteln zu ermöglichen.

```
100 REM -------------------------------
110 REM --- Farblauf an Rechtecken  ---
120 REM -------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
260 INK 0,0
270 :
280 REM -------------------------------
290 REM ---    Zeichnen der Figur    ---
300 REM -------------------------------
310 :
```

Programm 5.30: Farblauf an Rechtecken

```
320 FOR i=0 TO 200 STEP 4
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    MOVE i,i
360    DRAW i,400-i,farbe
370    DRAW 640-i*1.5,400-i,farbe
380    DRAW 640-i*1.5,i,farbe
390    DRAW i,i,farbe
400 NEXT
410 :
420 REM -------------------------------
430 REM --- 'Laufen' der Farben     ---
440 REM -------------------------------
450 :
460 FOR j=1 TO 100
470    FOR i=1 TO 15
480    INK i,farbe(((i+j) MOD 15)+1)
490    NEXT
500 NEXT
```

Programm 5.30: Farblauf an Rechtecken (Forts.)

Durch die Schleife ab Zeile 320 werden jeweils ineinandergeschachtelte Rechtecke ausgegeben. Durch das spätere „Laufen“ der Farben entsteht so der Eindruck, als würde man einen Gang entlangrennen. Auf einen Kontrollausdruck wurde an dieser Stelle bewußt verzichtet, da aus einem Hardcopy-Ausdruck nichts zu entnehmen wäre (der ganze Bildschirm ist gefüllt). Der Rest des Programms bedarf sicherlich keiner weiteren Erklärung.

Das gleiche Prinzip ist im folgenden Programmlisting verwendet (Programm 5.31), jedoch werden hier ineinandergeschachtelte Ellipsen gezeichnet.

```
100 REM -------------------------------
110 REM ---  Farblauf an Ellipsen   ---
120 REM -------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
```

Programm 5.31: Farblauf an Ellipsen

```
260 INK 0,0
270 :
280 REM ------------------------------
290 REM ---   Zeichnen der Ellipsen ---
300 REM ------------------------------
310 :
320 FOR i=8 TO 320 STEP 8
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    FOR j=0 TO 2*PI STEP (2*PI)/(i*1.5)
360       PLOT 320 + i*SIN(j),200 + i/2*COS(j),farbe
370    NEXT
380 NEXT
390 :
400 REM ------------------------------
410 REM --- 'Laufen' der Farben     ---
420 REM ------------------------------
430 :
440 FOR j=1 TO 100
450    FOR i=1 TO 15
460    INK i,farbe(((i+j) MOD 15)+1)
470    NEXT
480 NEXT
```

Programm 5.31: Farblauf an Ellipsen (Forts.)

Durch die Schrittweite (8) innerhalb der Programmschleife ab Zeile 320 bleibt noch ein Teil des Hintergrundes erhalten, was einen optisch besseren Effekt ergibt. Anders als beim Kreis-Unterprogramm in Kapitel 2 haben wir hier die punktförmige Ausgabe der Ellipse vorgesehen, was natürlich eine längere Wartezeit bis zur Fertigstellung der Grafik bedeutet.

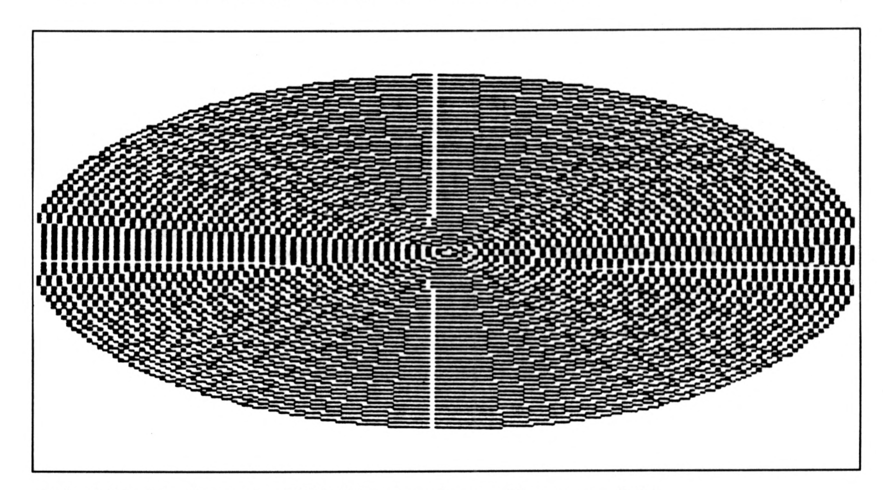


Abb. 5.30: Hardcopy der Bildschirmausgabe von Programm 5.31

```
100 REM -------------------------------
110 REM ---   Farblauf an Kegel     ---
120 REM -------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
260 INK 0,0
270 :
280 REM -------------------------------
290 REM ---    Zeichnen des Kegel   ---
300 REM -------------------------------
310 :
320 FOR i=0 TO 1.5*PI STEP 2*PI/128
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    MOVE 20*i,400-25*i
360    DRAW 360+240*SIN(i),160+140*COS(i),farbe
370 NEXT
380 :
390 REM -------------------------------
400 REM --- 'Laufen' der Farben     ---
410 REM -------------------------------
420 :
430 FOR j=1 TO 100
440    FOR i=1 TO 15
450    INK i,farbe(((i+j) MOD 15)+1)
460    NEXT
470 NEXT
```

Programm 5.32: Farblauf an einem Kegel (Version 1)

Wenden wir uns nun wieder den grafischen Figuren zu, denen Linien zugrunde liegen, deren Anfangs- und Endpunkt jeweils durch eine Schleife bestimmt ist (Programm 5.32).

Im Prinzip wird ein entarteter Kegel ausgegeben, wobei der Anfang der Linie (MOVE-Befehl) auf einer Geraden von oben links nach unten rechts läuft. Der Endpunkt der Linie durchläuft eine Ellipse, deren Koordinaten im DRAW-Befehl angegeben sind.

Im Gegensatz zum nächsten Beispiel wurde hier die relativ kleine Schrittweite 128 gewählt und die Schleife auch nur für einen ¾-Kreis durchlaufen. Beim nächsten Beispiel wird durch das Schleifenendekriterium in Zeile 320 ein biß-

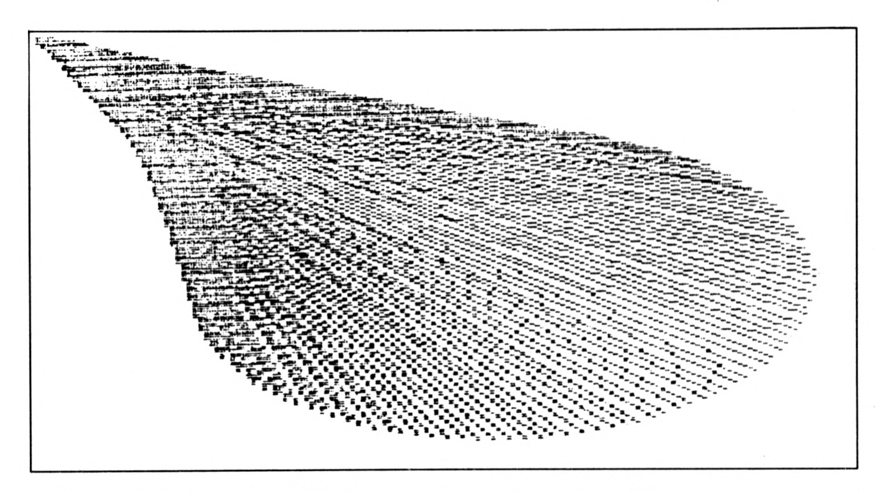


Abb. 5.31: Hardcopy der Bildschirmausgabe von Programm 5.32

chen mehr als ein Kreis beschrieben, und die „Auflösung“ ist auch größer, da die Schrittweite sehr klein ist (die 400 steht im Nenner).

Außerdem wurde die Linie, die die Anfangspunkte beschreibt, etwas mehr nach oben gezogen, was durch die Faktoren in Zeile 350 bewirkt wird. Zeile 360 wurde gegenüber dem letzten Beispiel nicht geändert.

```
100 REM -------------------------------
110 REM ---  Farblauf an Kegel      ---
120 REM -------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
260 INK 0,0
270 :
280 REM -------------------------------
290 REM ---   Zeichnen des Kegel    ---
300 REM -------------------------------
```

Programm 5.33: Farblauf an einem Kegel (Version 2)

```
310 :
320 FOR i=0 TO 2.1*PI STEP 2*PI/400
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    MOVE 80*i,400-50*i
360    DRAW 360+240*SIN(i),160-140*COS(i),farbe
370 NEXT
380 :
390 REM -------------------------------
400 REM --- 'Laufen' der Farben     ---
410 REM -------------------------------
420 :
430 FOR j=1 TO 100
440    FOR i=1 TO 15
450    INK i,farbe(((i+j) MOD 15)+1)
460    NEXT
470 NEXT
```

Programm 5.33: Farblauf an einem Kegel (Version 2) (Forts.)

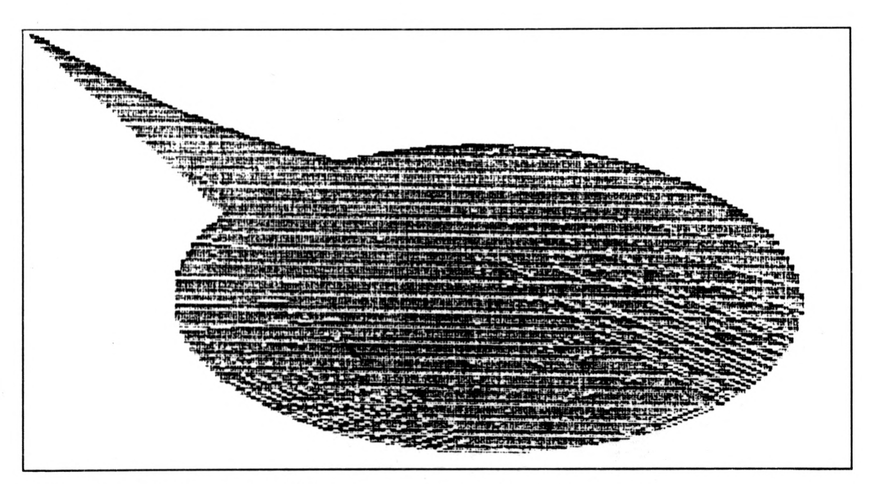


Abb. 5.32: Hardcopy der Bildschirmausgabe von Programm 5.33

```
100 REM -------------------------------
110 REM ---   Farblauf an Kegel     ---
120 REM -------------------------------
130 :
140 DIM farbe(16)
150 :
160 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
```

Programm 5.34: Farblauf an einem Kegel (Version 3)

```
170 :
180 FOR i=0 TO 15
190   READ farbe
200   INK i,farbe
210   farbe(i)=i
220 NEXT
230 :
240 MODE 0
250 :
260 INK 0,0
270 :
280 REM -------------------------------
290 REM ---    Zeichnen des Kegel    ---
300 REM -------------------------------
310 :
320 FOR i=0 TO 4*PI STEP PI/30
330    farbe=farbe+1
340    IF farbe=15 THEN farbe=1
350    MOVE 45*i,400-30*i
360    DRAW 280+200*SIN(i),200-120*COS(i),farbe
370 NEXT
380 :
390 REM -------------------------------
400 REM --- 'Laufen' der Farben      ---
410 REM -------------------------------
420 :
430 FOR j=1 TO 100
440    FOR i=1 TO 15
450    INK i,farbe(((i+j) MOD 15)+1)
460    NEXT
470 NEXT
```

Programm 5.34: Farblauf an einem Kegel (Version 3) (Forts.)

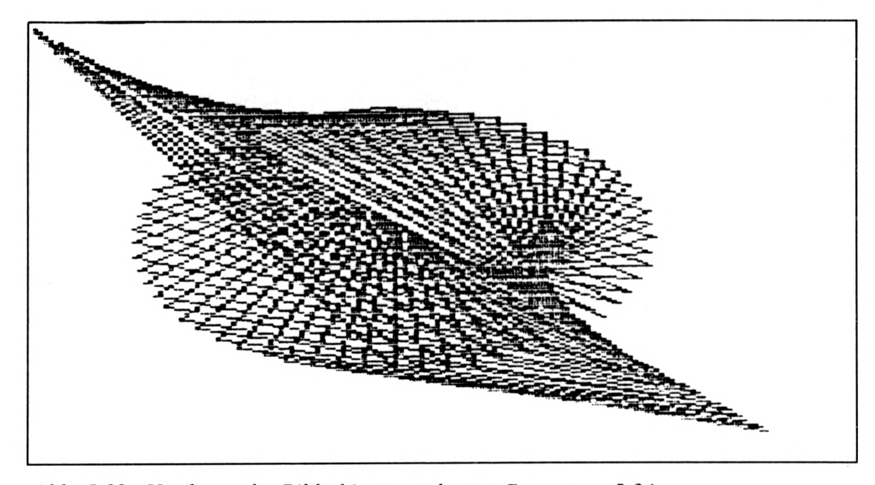


Abb. 5.33: Hardcopy der Bildschirmausgabe von Programm 5.34

Im nächsten Beispiel (Programm 5.34) wurde die Schrittweite nochmals verändert (sehr grobes Raster) und insbesondere das Schleifenendekriterium so gewählt, daß die Endpunkte der Linie zweimal den Kreis durchschreiben. Dadurch ergibt sich an den Bildschirmpunkten auf dem Kreisumfang der optische Eindruck eines Knicks in den Linien, obwohl zwei Linien nacheinander gezeichnet werden. Die Zeilen 350 und 360 wurden gegenüber dem letzten Listing ebenfalls wieder etwas geändert.

Abb. 5.33 gibt nur eine ungenügende Darstellung der Wirkung dieser Grafik am Bildschirm.

Auch an diesen Beispielen sieht man wieder, daß man durch geringfügige Änderung weniger Parameter das Aussehen von Grafiken total verändern kann.

Wir hoffen, Ihnen damit für diesen Bereich genügend Beispiele gegeben zu haben, um Ihre eigene „Grafikproduktion" etwas anzuregen.

5.3 FARBWECHSEL (TRANSPARENT-MODUS)

Nachdem wir nun ausführlich Beispiele zum Thema „laufende Farben" gebracht haben und Sie nebenbei einige Möglichkeiten zur Darstellung von Grafiken gelernt haben, wollen wir uns nun in den nächsten fünf Beispielen mit einem weiteren optischen Trick beschäftigen: dem Transparent-Modus.

Wie Sie wissen, können Sie bis zu sechzehn Farben gleichzeitig auf dem Bildschirm darstellen, wobei Sie beim Zeichnen über den Farbcode die Zahlen zwischen 0 und 15 angeben. Die Farben wurden vorher mittels des INK-Befehls aus der Farbpalette von 27 Farben ausgewählt. Wenn Sie nun nach dem Zeichnen einer Linie die der Linie zugeordnete Farbauswahlnummer beim INK-Befehl verändern, so verfärbt sich die Linie. Dies ist natürlich nicht nur mit einer einzelnen Farbe, sondern mit allen sechzehn Farben möglich.

Diesen Umstand wollen wir uns im folgenden zunutze machen, indem wir zunächst alle sechzehn Farbauswahlnummern (0 bis 15) mit der gleichen Farbe belegen. Dann werden wir nachträglich diese Farben ändern. Dabei treten einige interessante Effekte auf, wie Sie bereits dem ersten Listing (Programm 5.35) entnehmen können.

Mit diesem Programm werden nacheinander 500 willkürlich gewählte Linien auf dem Bildschirm in Weiß dargestellt. Die Nummer der aktuell bezeichneten Linie können Sie in der ersten Zeile ablesen. Sobald sich die Nummer der Zahl 500 nähert, sollten Sie aufmerksam zusehen: Nacheinander färben sich die Linien in Gruppen um, bis keine Linie mehr in Weiß dargestellt ist. Dann werden alle Linien nochmals mit einer anderen Farbe versehen, der Hintergrundfarbe, bis die Darstellung „verschwunden" ist.

```
100 REM -----------------------------
110 REM --- Linien im Transparent-  ---
120 REM ---                   modus ---
130 REM -----------------------------
140 :
150 MODE 0
160 :
170 FOR i=1 TO 15
180     INK i,26
190 NEXT
200 :
210 RANDOMIZE 464
220 :
230 FOR i=1 TO 500
240     zufall1 = INT(640*RND(640))
250     zufall2 = INT(400*RND(200))
260     zufall3 = INT(15*RND(15))+1
270     zufall4 = INT(640*RND(640))
280     zufall5 = INT(400*RND(200))
290     MOVE zufall1,zufall2
300     DRAW zufall4,zufall5,zufall3
310     LOCATE 1,1
320     PRINT"Linie ";i
330 NEXT
340 :
350 FOR i=1 TO 15
360     INK i,i
370     t=TIME
380     IF TIME-t < 30 THEN 380
390 NEXT
400 :
410 FOR i=1 TO 15
420     INK i,1
430     t=TIME
440     IF TIME-t < 60 THEN 440
450 NEXT
```

Programm 5.35: Linien im Transparent-Modus

Das Ergebnis eines Programmlaufs sehen Sie – allerdings nur in Schwarzweiß – in Abb. 5.34.

Im Programmlisting wird zunächst der 16-Farben-Modus eingeschaltet. Dann werden die Farbauswahlnummern 1 bis 15 mit der Farbe Leuchtendweiß besetzt. In Zeile 210 wird der Zufallsgenerator initialisiert, wodurch in den Zeilen ab 240 innerhalb der Schleife von Zeile 230 bis Zeile 330 fünf verschiedene Zufallszahlen ausgewählt werden. Je zwei Zufallszahlen befinden sich im Bereich zwischen 0 und 639 und zwischen 0 und 399. Die Variable zufall3 nimmt Werte zwischen 1 und 15 an. Mit Hilfe dieser Zufallsvariablen werden die Linien gezogen, wobei in den Zeilen 310 und 320 die Ausgabe der aktuellen Liniennummer in der ersten Zeile erfolgt.

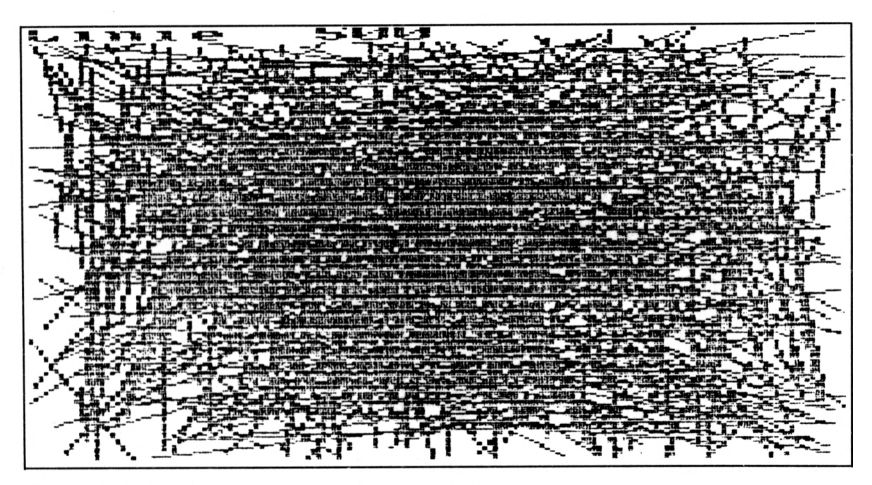


Abb. 5.34: Linien im Transparent-Modus, Hardcopy

In den Zeilen ab 350 werden nacheinander alle Farbauswahlnummern mit der Farbe gleicher Nummer besetzt. Zeilen 370 und 380 bilden eine Warteschleife, die von Ihnen nach Ihren Wünschen angepaßt werden kann.

Sofern Sie das Programm nach dem Einschalten oder einem Reset des Computers gestartet haben, ist für die Farbauswahlnummer 0 die Farbe 1 vorgesehen (Blau). In den Zeilen ab 410 wird analog zur vorangegangenen FOR...NEXT-Schleife ebenfalls auf Blau gesetzt.

Im nächsten Listing (Programm 5.36) haben wir das gleiche Verfahren angewendet, nur daß wir nicht die Linien zufällig auswählen, sondern den gesamten Bildschirm mit senkrechten Linien füllen, die zunächst auch weiß sind.

```
100 REM ------------------------------
110 REM --- Senkrechte Linien im    ---
120 REM ---          Transparentmodus ---
130 REM ------------------------------
140 :
150 :
160 MODE 0
170 :
180 FOR i=1 TO 15
190     INK i,26
200 NEXT
210 :
220 FOR i=0 TO 639 STEP 4
```

Programm 5.36: Senkrechte Linien im Transparent-Modus

```
230     MOVE i,0
240     DRAW i,400,i/4 MOD 14 + 1
250 NEXT
260 :
270 FOR i=2 TO 15
280     INK i,i
290     FOR j=i+1 TO 15
300        INK j,i
310        REM GOTO 340
320     t=TIME
330     IF TIME-t < 90 THEN 330
340     NEXT
350     REM GOTO 380
360     t=TIME
370     IF TIME-t < 120 THEN 370
380 NEXT
390 :
400 FOR i= 15 TO 1 STEP -1
410     INK i,1
420     t=TIME
430     IF TIME-t < 90 THEN 430
440 NEXT
450 :
460 GOTO 270
```

Programm 5.36: Senkrechte Linien im Transparent-Modus (Forts.)

Das Zeichnen der senkrechten Linien erfolgt in der FOR...NEXT-Schleife in den Zeilen 220 bis 250. Durch die REM-Befehle in den Zeilen 310 und 350 wird die Warteschleife übergangen. Besonders der Zeitablauf ist für den Effekt sehr wichtig, so daß Sie hier unterschiedliche Wartezeiten ausprobieren sollten. Auch hier werden die weißen Linien nacheinander wieder anders eingefärbt (FOR...NEXT-Schleife zwischen den Zeilen 270 und 380) und anschließend wieder auf die Hintergrundfarbe gesetzt (FOR...NEXT-Schleife ab Zeile 400).

Durch den Sprungbefehl in Zeile 460 wiederholt sich das Schauspiel fortlaufend.

Anders als im letzten Beispiel werden hier aber jeweils Bereiche immer neu umgefärbt, und zwar in der Art, daß der eingefärbte Streifen immer schmaler wird. Schauen Sie sich die Wirkung am besten selbst auf dem Bildschirm an. Auf eine Hardcopy haben wir hier verzichtet, da daraus sowieso nichts zu ersehen wäre.

Im nächsten Beispiel (Programm 5.37) haben wir Ihnen durch die INPUT-Befehle die Möglichkeit vorgegeben, die beiden wichtigsten Farben und die Wartezeiten beim Programmstart selbst zu bestimmen. Außerdem benutzen wir wieder ein Feld farbe(), in dem wir vordefinierte Farben verwenden. Die sich

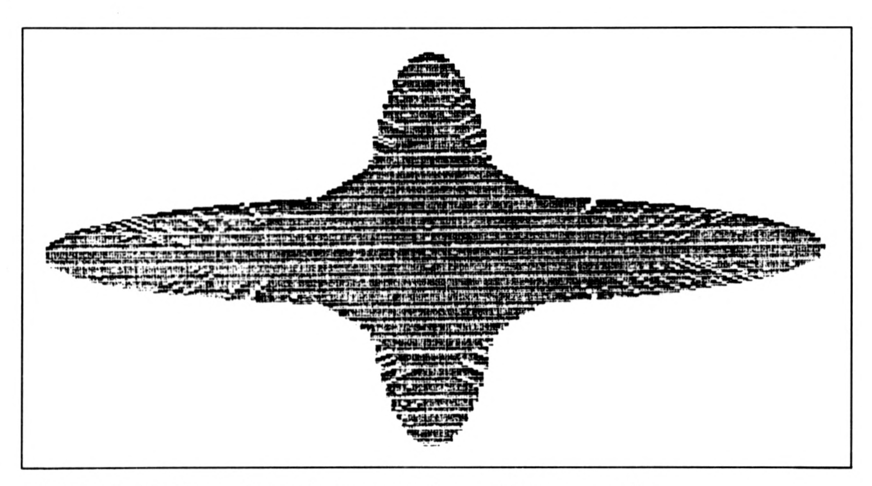


Abb. 5.35: Hardcopy der Bildschirmausgabe von Programm 5.37

daraus ergebende Programmumstellung dürfte Ihnen aus den letzten Kapiteln schon bekannt sein. Wir benutzen zum Zeichnen wieder Linien, die zwischen Ellipsen liegen. Siehe dazu auch Abb. 5.35.

```
100 REM -------------------------------
110 REM --- Linien zwischen Ellipsen---
120 REM --- im Transparentmodus     ---
130 REM -------------------------------
140 :
150 DATA 26,25,24,22,20,18,17,14,13,11,8,7,6,5,2,1
160 :
170 INPUT "farbe 1";farbel
180 INPUT "farbe 2";farbe2
190 INPUT "Wartezeit 1";wartel
200 INPUT "Wartezeit 2";warte2
210 :
220 DIM farbe(16)
230 :
240 FOR i=0 TO 15
250   READ farbe
260   farbe(i)=i
270 NEXT
280 :
290 INK 0,farbel
300 :
310 MODE 0
320 :
330 FOR i=1 TO 15
```

Programm 5.37: Linien zwischen Ellipsen, Transparent-Modus

```
340     INK i,26
350 NEXT
360 :
370 FOR i=0 TO 2*PI STEP 2*PI/400
380     farbe=farbe+1
390     IF farbe=15 THEN farbe=1
400     MOVE 320-300*SIN(i),200 - 50*COS(i)
410     DRAW 320+ 50*SIN(i),200 + 180*COS(i),farbe(farbe
)
420 NEXT
430 :
440 FOR i=2 TO 15
450     INK i,i
460 :
470     FOR j=i+1 TO 15
480         INK j,i
490         t=TIME
500         IF TIME-t < wartel THEN 500
510     NEXT
520 :
530     t=TIME
540     IF TIME-t < 300 THEN 540
550 NEXT
560 :
570 FOR i= 15 TO 1 STEP -1
580     INK i,farbe2
590     t=TIME
600     IF TIME-t < 90 THEN 600
610 NEXT
620 :
630 GOTO 440
```

Programm 5.37: Linien zwischen Ellipsen, Transparent-Modus (Forts.)

Achten Sie auf die relativ geringe Schrittweite beim Zeichnen der Grafik in Zeile 370. Auch sie ist für den auftretenden Effekt entscheidend verantwortlich.

Probieren Sie z. B. die Farbkombinationen 0,0 und verschiedene Wartezeiten aus. Auch andere der bisher vorgestellten Figuren sollten Sie im Hinblick auf ihre Eignung zur Verwendung des Transparent-Modus prüfen und gegebenenfalls ausprobieren.

Natürlich muß man nicht alle Farbauswahlnummern nacheinander mit anderen Farben belegen. Im folgenden wollen wir Ihnen ein kurzes Beispiel (Programm 5.38) aufzeigen, wo lediglich eine Farbe verwendet wird, aber durch die umlaufende Zuweisung ein weiterer besonderer Effekt auftritt: Die Figur scheint sich zu bewegen.

Gezeichnet werden 15 „Räder“, wie sie in Abb. 5.36 dargestellt sind. Sie bestehen jeweils aus einem Kreis mit sechzehn eingezeichneten Radien.

```
100 REM ------------------------------
110 REM --- Ein sich drehendes Rad  ---
120 REM ------------------------------
130 :
140 MODE 0
150 :
160 INK 0,0
170 PAPER 0
180 :
190 FOR i=1 TO 15
200   INK i,8
210 NEXT
220 :
230 farbe=1
240 x=320
250 y=200
260 abstand=100
270 :
280 FOR j= 1 TO 15
290    weiter = j*2*PI/255
300    GOSUB 500
310    farbe=j
320 NEXT
330 :
340 FOR i=1 TO 10000
350    ganzalt=alt
360    alt=farbe
370    farbe = farbe+1
380    IF farbe=16 THEN farbe=1
390    INK farbe,6
400    INK ganzalt,0
410 NEXT
420 :
430 PAPER 0
440 INK 0,0
450 PEN 1
460 INK 1,26
470 :
480 END
490 :
500 REM ------------------------------
510 REM ---      Zeichnen des Rades ---
520 REM ------------------------------
530 :
540    MOVE x,y
550 :
560    FOR i=0 TO 2*PI STEP 2*PI/16
570       kx=x+abstand*SIN(i+weiter)
580       ky=y+abstand*COS(i+weiter)
590       DRAW kx,ky,farbe
600       MOVE x,y
610       DRAW kx,ky,farbe
620    NEXT
630 :
640 RETURN
```

Programm 5.38: Drehendes Rad

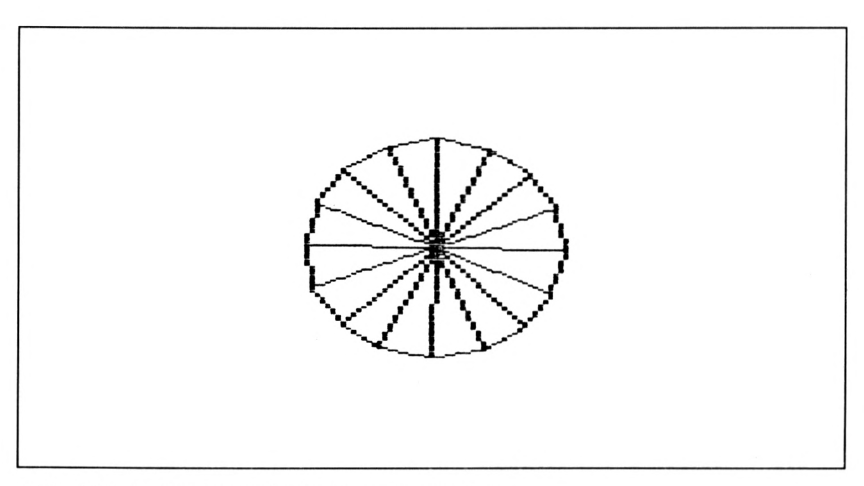


Abb. 5.36: Senkrechte Linien im Transparent-Modus

Die einzelnen Räder werden so ausgegeben, daß sie zusammengezeichnet fast einen Vollkreis ergeben, wie er in Abb. 5.37 dargestellt ist.

Bei unserem Programm ist das Zeichnen dieser Räder am Bildschirm direkt dargestellt, diesmal also nicht unter Verwendung des Transparent-Modus. Wollen Sie eventuelle Zuschauer verblüffen, so können Sie das Zeichnen auch in der gleichen Farbe vornehmen, die Sie für den Hintergrund gewählt haben.

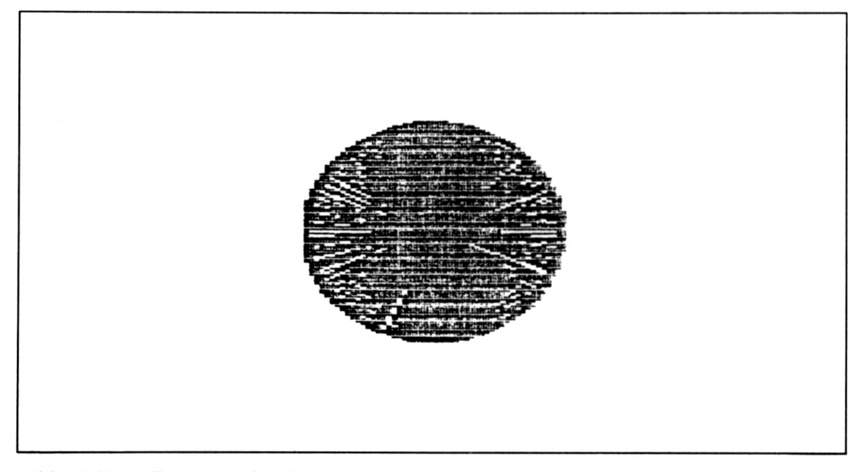


Abb. 5.37: Alle Linien des drehenden Rades auf einmal sichtbar gemacht

Durch die Zuordnung in den Zeilen 160 und 170 wird der Hintergrund schwarz eingefärbt. Dann wird für die Farbauswahlnummern 1 bis 15 jeweils die Farbe 8 vorgesehen. Ab Zeile 230 werden die Parameter für das Rad festgelegt, wobei wir von einer Verwendung einer Ellipse abraten.

Über die Variable weiter wird der jeweilige Winkel weitergeschaltet, was die Überlappung ergibt.

Den eigentlichen Effekt bergen die Programmzeilen ab 340 in sich. Um das „Laufen" des Rades fließend zu gestalten, werden die Variablen ganzalt, alt und farbe verwendet, wobei die Variable farbe hier nicht eine Farbnummer, sondern eine Farbauswahlnummer im INK-Befehl angibt (siehe Zeile 390). Diese Variable wird jeweils weitergeschaltet und bei Erreichen des Wertes 16 entsprechend auf den Wert 1 zurückgeschaltet. Der Wert wird bei jedem neuen Durchlauf an die Variable alt übergeben, deren Wert dann in der Variablen ganzalt zu finden ist. Nachdem ein weiteres Rad mit der Farbe 6 (Hellrot) eingefärbt wurde, wird die älteste Einfärbung eines Rades wieder auf die Hintergrundfarbe geschaltet.

Zum Zeichnen des Rades wurde das Unterprogramm ab Zeile 500 verwendet, dessen Erläuterungen Sie bitte dem Kapitel 2 entnehmen.

Abschließend wollen wir Ihnen noch ein kleines Beispielprogramm vorstellen, an dem Sie Ihrer Phantasie vollkommen freien Lauf lassen sollten (Programm 5.39). Es werden 8speichige Räder in zwei verschiedenen Größen und zwei verschiedenen Farben am Bildschirm dargestellt. Den Testausdruck finden Sie in Abb. 5.38.

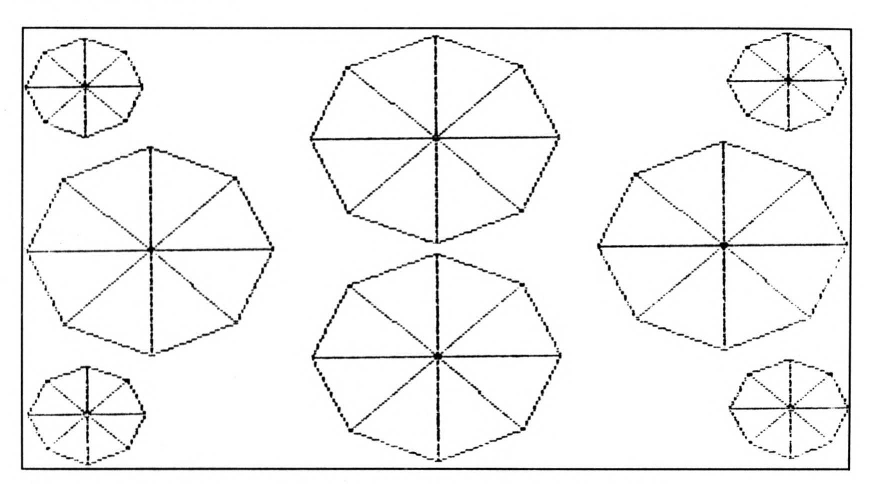


Abb. 5.38: Hardcopy der Bildschirmausgabe von Programm 5.39

```
100 REM ------------------------------
110 REM --- Beispielprogramm zum    ---
120 REM ---                Aendern ---
130 REM ------------------------------
140 :
150 CLS
160 :
170 INK 1,6
180 INK 2,15
190 :
200 farbe=1
210 :
220 x=50 : y=50 : Abstand=45 : GOSUB 540
230 x=590 : y=50 : Abstand=45 : GOSUB 540
240 x=590 : y=350 : Abstand=45 : GOSUB 540
250 x=50 : y=350 : Abstand=45 : GOSUB 540
260 :
270 farbe=2
280 :
290 x=100 : y=200 : Abstand=95 : GOSUB 540
300 x=540 : y=200 : Abstand=95 : GOSUB 540
310 x=320 : y=100 : Abstand=95 : GOSUB 540
320 x=320 : y=300 : Abstand=95 : GOSUB 540
330 :
340 END
350 :
500 REM ------------------------------
510 REM --- Unterprogramm fuer Figur---
520 REM ------------------------------
530 :
540    MOVE x,y
550 :
560    FOR i=0 TO 2*PI STEP 2*PI/8
570       kx=x+abstand*SIN(i)
580       ky=y+abstand*COS(i)
590       DRAW kx,ky,farbe
600       MOVE x,y
610       DRAW kx,ky,farbe
620    NEXT
630 :
640 RETURN
```

Programm 5.39: 8speichige Räder

Sie sollten nun die vorgestellten Effekte auf die verschiedenen Räder anwenden. Viel Spaß!

5.4 VERSCHIEDENE ZEICHENMODI

Wenn Sie mit den Grafikbefehlen eine Linie ziehen oder einen Punkt ausgeben, sind Sie gewohnt, daß dieser Punkt in der angegebenen Farbe auf dem Bildschirm erscheint. Dies ist auch der Normalzustand Ihres CPC 464 nach dem Einschalten.

Es gibt aber noch drei verschiedene andere Modi, die für spezielle Effekte genutzt werden können. Diese Effekte müssen nicht unbedingt nur künstlerisch oder spielerisch genutzt werden, sondern können auch bei kommerziellen Grafiken angewendet werden, da sie unter anderem das Löschen einer gezeichneten Figur erlauben, ohne daß der Hintergrund dadurch berührt wird. In diesem Kapitel haben wir generell auf Hardcopys verzichtet, da die verwendeten Abbildungen lediglich den Zweck der Verdeutlichung haben sollen, ohne daß ein Anspruch auf optische Qualität besteht.

Die verschiedenen Modi können beim CPC 464 entweder mit Steuerzeichen oder über den CHR$()-Befehl eingeschaltet werden, beim CPC 664 und CPC 6128 kann der Modus direkt bei den Grafikbefehlen DRAW, DRAWR, MOVE, MOVER, INK, PEN, PLOT und PLOTR angegeben werden. Die Änderung des Schreibmodus wird beim 464 mit zwei Werten erreicht, wobei der erste Wert konstant 23 ist und die Steuerinformation beinhaltet und der zweite Werte zwischen 0 und 3 annehmen kann und den gewünschten Modus beinhaltet.

Für den zweiten Parameter gilt folgende Tabelle:

0 – Normalmodus (Schreiben)
1 – XOR-Modus
2 – AND-Modus
3 – OR-Modus

Die Modi arbeiten direkt mit der bitweisen Darstellung auf dem Bildschirm, und so ist es nicht verwunderlich, daß die Modi auch mit den möglichen logischen Verknüpfungen, wie sie bei Bitmustern Anwendung finden, bezeichnet werden.

Verknüpft werden jeweils die Bitmuster für die derzeit am Bildschirm ersichtliche Farbe und die zu zeichnende Farbe. Da bis zu sechzehn Farben möglich und diese in vier Bits darstellbar sind, wollen wir die folgenden Beispiele auch auf vier Bits beschränken:

0101	0101	0101
XOR 1100	AND 1100	OR 1100
1001	0100	1101

Für die Farbauswahlnummern würde dies bedeuten:

5	5	5
XOR 12	AND 12	OR 12
9	4	13

Für diejenigen, die in der Bit-Manipulation nicht so bewandert sind, hier noch eine kurze Erklärung:

AND/UND-Verknüpfung

Bei der UND-Verknüpfung ist das Ergebnis genau dann ein gesetztes Bit, wenn sowohl am Bildschirm als auch in der zu zeichnenden Farbe das entsprechende Bit gesetzt ist.

OR/ODER-Verknüpfung

Bei der ODER-Verknüpfung ist genau dann ein Bit gesetzt, wenn bei der am Bildschirm befindlichen Farbe das entsprechende Bit gesetzt ist oder das entsprechende Bit in der zu zeichnenden Farbe gesetzt ist. In diesem Falle haben wir es mit einem einschließlichen ODER zu tun, d. h. das Bit ist auch gesetzt, wenn beide Ausgangsbits gesetzt sind.

XOR/EXKLUSIV ODER-Verknüpfung

Im Unterschied zur OR/ODER-Verknüpfung ist hier das Bit nur gesetzt, wenn entweder das eine oder das andere Bit gesetzt ist. Der Unterschied besteht hier in der Ausschließlichkeit, d. h. es darf ausschließlich eins der beiden Ausgangsbits gesetzt sein, und somit ist bei beiden gesetzten Ausgangsbits das Ergebnis ein nicht gesetztes Bit.

Hier eine ausführliche Tabelle, auch Wertetabelle genannt:

A	B	A XOR B	A AND B	A OR B
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

Im ersten Programmbeispiel wollen wir die Wirkung am Bildschirm veranschaulichen (Programm 5.40).

Da wir nur vier mögliche Zeichenmodi haben, genügt uns hier der 4-Farben-Modus. Dann wählen wir uns ab Zeile 160 vier Farben aus, wobei der Bildschirmhintergrund auf Farbe 2 gesetzt wird. Das Einfärben erreichen wir durch Zeile 220.

In Zeile 260 und Zeile 270 ziehen wir eine Linie von der Ecke unten links zur Ecke oben rechts und wählen dabei die Farbauswahlnummer 1 mit Farbe 6 (Hellrot). Nach dem Programmstart ist auch eine hellrote Linie am Bildschirm ersichtlich. Die Zeilen 290 und 300 stellen nur eine Warteschleife auf Tasten-

druck dar, damit Sie das Geschehen am Bildschirm auch nachvollziehen können.

```
100 REM ------------------------------
110 REM --- Verschiedene Zeichenmodi---
120 REM ------------------------------
130 :
140 MODE 1
150 :
160 INK 0,0
170 INK 1,6
180 INK 2,15
190 INK 3,26
200 :
210 PAPER 2
220 CLS
230 :
240 REM ---          normal          ---
250 :
260 MOVE 0,0
270 DRAW 639,399,1
280 :
290 a$=INKEY$
300 IF a$="" THEN 290
310 :
320 REM ---      XOR-Modus           ---
330 :
340 PRINT CHR$(23);CHR$(1)
350 :
360 MOVE 0,399
370 DRAW 639,0,1
380 :
390 a$=INKEY$
400 IF a$="" THEN 390
410 :
420 REM ---      AND-Modus           ---
430 :
440 PRINT CHR$(23);CHR$(2)
450 :
460 MOVE 320,0
470 DRAW 320,399,1
480 :
490 a$=INKEY$
500 IF a$="" THEN 490
510 :
520 REM ---      OR-Modus            ---
530 :
540 PRINT CHR$(23);CHR$(3)
550 :
560 MOVE 0,200
570 DRAW 639,200,1
580 :
590 REM --- Zurueckschalten auf      ---
600 REM ---                   normal ---
610 :
620 PRINT CHR$(23);CHR$(0)
```

Programm 5.40: Verschiedene Zeichenmodi

In Zeile 340 wird auf den XOR-Modus umgeschaltet. Bei Besitzern eines CPC 664 oder 6128 kann Zeile 340 entfallen. In diesem Fall muß allerdings Zeile 360 wie folgt lauten:

```
360 MOVE 0,399,,1
```

wobei die letzte Ziffer hier den Schreibmodus, entsprechend der vorgenannten Tabelle, bestimmt.

Ebenso kann Zeile 440 weggelassen werden, und Zeile 460 ist nach obigem Schema zu ändern.

Auch bei den weiteren Programmen in diesem Kapitel können Besitzer eines 664 oder 6128 diese Änderungen leicht durchführen, ohne daß wir nochmals darauf eingehen werden.

Trotzdem laufen auf den beiden neueren CPC-Rechnern auch die Programme in der vorgegebenen Form. Mit Rücksicht auf die 464-Benutzer haben wir deshalb die allgemein gültige Form der Ausgabe gewählt. In diesem Falle ziehen wir die Linie von der Ecke oben rechts zur Ecke unten links. Auch hier wählen wir wiederum die Farbauswahlnummer 1 mit der entsprechenden Farbe Hellrot, aber das Ergebnis am Bildschirm ist eine weiße Linie. Dies ist ganz natürlich, wenn Sie sich obige Beispiele anschauen. Hier noch einmal der konkrete Fall als Bitmuster:

Bitmuster für den Hintergrund: 10 (dezimal 2)
Bitmuster für die Zeichenfarbe: 01 (dezimal 1)

Da in beiden Fällen (die beiden linken Bits und die beiden rechten Bits) jeweils ein Bit gesetzt ist, ist das Ergebnis einer EXKLUSIV ODER-Verknüpfung das Bitmuster 11 (dezimal 3). Wie wir in Zeile 190 sehen, ist für die Farbauswahlnummer 3 auch die Farbe 26 (Leuchtendweiß) vorgesehen.

Als nächstes ziehen wir in den Zeilen 460 und 470 eine senkrechte Linie in der Bildschirmmitte, ebenfalls unter Angabe der Farbauswahlnummer 1. Das Ergebnis ist eine schwarze Linie. Auch dies ist ganz klar, da die Bitmuster 10 und 01 UND-verknüpft das Bitmuster 00 ergeben. Die Farbauswahlnummer 0 wurde auch mit der Farbe 0 (Schwarz) in Zeile 160 besetzt.

Als letztes wird der OR-Modus eingeschaltet, was in Zeile 540 passiert. In den Zeilen 560 und 570 wird eine waagerechte Linie in Bildschirmmitte gezeichnet. Auch hier ist das Ergebnis wieder eine weiße Linie, da sich bei Verknüpfung von 10 und 01 das Ergebnis nicht von der XOR-Verknüpfung unterscheidet. Lassen Sie nun das ganze Programm laufen, indem Sie in den Zeilen 370, 470 und 570 als Farbauswahlnummer jeweils eine 2 angeben.

Diesmal ist die Diagonale von oben links nach unten rechts schwarz, und die waagerechte sowie die senkrechte Linie erscheinen gar nicht. Daß sie trotzdem gezeichnet werden, stellen Sie bei genauem Hinsehen exakt in der Bildschirmmitte fest.

Warum dies so ist, läßt sich leicht beim Nachrechnen der Bitmuster feststellen.

Sofern Sie an Ihrem Computer nach dem Programm keinen Reset durchführen, ist Zeile 620 sehr wichtig, da sie den Rechner wieder in den Normalmodus zurückschaltet.

Im nächsten Beispiel (Programm 5.41) verwenden wir den XOR-Modus, um darzustellen, daß mit seiner Hilfe das Überschreiben und wieder Löschen eines Bereichs ohne Zerstörung des Hintergrunds erfolgen kann.

```
100 REM -------------------------------
110 REM --- Verwendung von XOR zum  ---
120 REM ---   Loeschen einer Figur  ---
130 REM -------------------------------
140 :
150 MODE 1
160 :
170 INK 0,0
180 INK 1,6
190 INK 2,15
200 INK 3,26
210 :
220 :
230 REM ---         normal          ---
240 :
250 FOR i=160 TO 480 STEP 8
260    MOVE i,300
270    DRAW i,100,1
280 NEXT
290 :
300 REM --- Abwechselndes Zeichnen  ---
310 REM ---        und Loeschen     ---
320 :
330 PRINT CHR$(23);CHR$(1)
340 :
350 FOR i=240 TO 400 STEP 8
360    MOVE i,250
370    DRAW i,150,2
380 NEXT
390 :
400 zaehl=zaehl+1
410 LOCATE 3,3
420 PRINT"Zaehler: ";zaehl
430 :
440 a$=INKEY$
450 IF a$="" THEN 440
460 GOTO 350
```

Programm 5.41: Zeichnen und wieder Löschen einer Figur mit Hilfe des XOR-Modus, Beispiel 1

Gezeichnet werden zwei Schraffuren, die erste in Hellrot, was in den Zeilen 250 bis 280 passiert. Dann wird der Zeichenmodus umgeschaltet und eine kleinere Schraffur in die größere eingelagert. Zur Überprüfung durch den Anwender wird noch ein Zähler fortgeschaltet und jeweils in der oberen linken Ecke am Bildschirm angezeigt. Nachdem eine Taste gedrückt wurde, wird wieder zum Zeichnen der kleineren Schraffur übergegangen.

Wenn Sie nun das Programm laufen lassen, werden Sie innerhalb der umgebenden hellroten Schraffur jeweils bei einem geraden Zählerstand keine weitere Schraffur feststellen, wohingegen bei einem ungeraden Zählerstand eine weiße Schraffur eingelagert ist.

Ein komplexeres Beispiel ist in Programm 5.42 dargestellt.

```
100 REM ------------------------------
110 REM --- Verwendung von XOR zum  ---
120 REM ---   Loeschen einer Figur  ---
130 REM ------------------------------
140 :
150 MODE 1
160 :
170 INK 0,0
180 INK 1,6
190 INK 2,15
200 INK 3,26
210 :
220 :
230 REM ---          normal          ---
240 :
250 PRINT CHR$(23);CHR$(0) : REM 'normal' sicherstellen
260 FOR i=160 TO 480 STEP 8
270    MOVE i,300
280    DRAW i,100,1
290 NEXT
300 :
310 FOR i=180 TO 340 STEP 2
320    MOVE i,280
330    DRAW i,180,2
340 NEXT
350 :
360 FOR i=300 TO 460 STEP 4
370    MOVE i,220
380    DRAW i,120,3
390 NEXT
400 :
410 FOR i=102 TO 538 STEP 8
420    MOVE i,380
430    DRAW i,20,0
440 NEXT
450 :
460 REM ---   Einschalten  XOR       ---
```

Programm 5.42: Zeichnen mit XOR-Modus, Beispiel 2

```
470 :
480 PRINT CHR$(23);CHR$(1)
490 :
500 FOR i=102 TO 538 STEP 8
510    MOVE i,380
520    DRAW i,20,1
530 NEXT
540 :
550 FOR i=300 TO 460 STEP 4
560    MOVE i,220
570    DRAW i,120,3
580 NEXT
590 :
```

Programm 5.42: Zeichnen mit XOR-Modus, Beispiel 2 (Forts.)

Diesmal sollten Sie selber herausfinden, was wann passiert.

Mit dem OR-Modus lassen sich sehr einfach Doppelbelegungen hervorheben, wie Programm 5.43 zeigt.

```
100 REM -------------------------------
110 REM --- Verwendung von OR zum   ---
120 REM --- Hervorheben von         ---
130 REM --- Doppelbelegungen        ---
140 REM -------------------------------
150 :
160 MODE 1
170 :
180 INK 0,0
190 INK 1,6
200 INK 2,15
210 INK 3,26
220 :
230 :
240 REM ---  OR-Modus einschalten   ---
250 :
260 PRINT CHR$(23);CHR$(3)
270 :
280 FOR i=100 TO 540 STEP 2
290    MOVE i,350
300    DRAW i,250,1
310 NEXT
320 :
330 FOR i=180 TO 460 STEP 2
340    MOVE i,280
350    DRAW i,180,2
360 NEXT
370 :
380 FOR i=50 TO 200 STEP 2
```

Programm 5.43: Hervorheben von Doppelbelegungen mit Hilfe des OR-Modus

```
390     MOVE i,300
400     DRAW i,120,3
410 NEXT
420 :
430 FOR i=100 TO 500 STEP 2
440     MOVE i,380
450     DRAW i,20,0
460 NEXT
470 :
```

Programm 5.43: Hervorheben von Doppelbelegungen mit Hilfe des OR-Modus (Forts.)

Es werden damit drei Blöcke gezeichnet, die sich gegenseitig überlappen. Damit werden die Grenzen dieser Vorgehensweise deutlich, da wir nur vier Farben zur Verfügung haben und eine für den Hintergrund reserviert sein sollte.

Der erste Block wird in der Mitte der oberen Bildschirmhälfte in Hellrot gezeichnet. In die Mitte darunter – etwas überlappend – wird ein oranger Block gezeichnet. Dort, wo sich die beiden Blöcke überlappen, ist der Bildschirm leuchtendweiß eingefärbt. Eine Überlappung des dritten Blocks (ab Zeile 380) mit den beiden ersten Blöcken ist nicht mehr ersichtlich, da hier bereits die Farbe Leuchtendweiß vorgesehen ist.

Für den praktischen Gebrauch ist der 4-Farben-Modus nur bei der Verwendung von zwei Blöcken oder ähnlichen Figuren interessant. Durch die ODER-Verknüpfung werden immer mehr Bits zur Bilddarstellung gesetzt, so daß bei häufiger Überlappung kein zusätzliches Bit mehr gesetzt werden kann.

Für einen optisch eindrucksvollen Aufbau sollten für höhere Farbauswahlnummern (d. h. Kombinationen mit mehreren Bits) hellere Farben vorgesehen werden. Auch sollte die Möglichkeit der Überlappung von vornherein bei Vergebung der Farbauswahlnummern beim Zeichnen berücksichtigt werden, so daß es später nicht zu ungewünschten Kollisionen kommen kann. Für eine sauber aufgebaute Grafik ist etwas Bitfummelei nicht zu vermeiden.

Das nächste Beispiel zeigt, daß man bei Verwendung von 16 Farben bereits einen Teil der Nachteile vermeiden kann. Wenn Sie sich das Bild am Bildschirm direkt anschauen, werden Sie feststellen, daß auch hier zwei verschiedene Bereiche der Überlappung mit der gleichen Farbe eingefärbt sind (Programm 5.44).

Ein kleiner Nebeneffekt am Rande: Versuchen Sie, in Zeile 280 als Modus eine 2 im zweiten Parameter einzugeben. Das Ergebnis? Nichts!

Den AND-Modus kann man also nicht zur Kennzeichnung von Doppelbelegungen verwenden, aber er läßt sich sehr gut zum Löschen einzelner Bild-

```
100 REM ------------------------------
110 REM --- Verwendung von OR zum    ---
120 REM --- Hervorheben von          ---
130 REM --- Doppelbelegungen         ---
140 REM ------------------------------
150 :
160 MODE 0
170 :
180 INK 0,0
190 INK 1,6
200 INK 2,15
210 INK 3,26
220 INK 4,4
230 INK 5,9
240 INK 6,11
250 :
260 REM ---  OR-Modus einschalten    ---
270 :
280 PRINT CHR$(23);CHR$(3)
290 :
300 FOR i=100 TO 540 STEP 4
310    MOVE i,350
320    DRAW i,250,1
330 NEXT
340 :
350 FOR i=180 TO 460 STEP 4
360    MOVE i,280
370    DRAW i,180,2
380 NEXT
390 :
400 FOR i=50 TO 200 STEP 4
410    MOVE i,300
420    DRAW i,120,3
430 NEXT
440 :
450 FOR i=100 TO 500 STEP 4
460    MOVE i,380
470    DRAW i,20,0
480 NEXT
490 :
500 FOR i=100 TO 500 STEP 4
510    MOVE i,200
520    DRAW i,50,5
530 NEXT
540 :
550 FOR i=400 TO 600 STEP 4
560    MOVE i,350
570    DRAW i,100,6
580 NEXT
```

Programm 5.44: Verwendung des OR-Modus zur Hervorhebung von Doppelbelegungen, MODE 0

schirmbereiche einsetzen, obwohl die dargestellte Grafik auch Möglichkeiten der Überlappung zeigt.

Wollen Sie z. B. das linke obere Viertel des Bildschirms löschen, so schalten

Sie zunächst den AND-Modus ein und zeichnen einen Block von den Koordinaten (0,399) bis (320,200) mit der Farbauswahlnummer 0.

```
100 REM -------------------------------
110 REM --- Verwendung von AND zum  ---
120 REM --- Loeschen von Doppel-    ---
130 REM --- belegungen              ---
140 REM -------------------------------
150 :
160 MODE 0
170 :
180 INK 0,0
190 INK 1,6
200 INK 2,15
210 INK 3,26
220 INK 4,4
230 INK 5,9
240 INK 6,11
250 INK 15,21
260 :
270 PAPER 15 : CLS
280 REM ---  OR-Modus einschalten   ---
290 :
300 PRINT CHR$(23);CHR$(2)
310 :
320 FOR i=100 TO 540 STEP 4
330    MOVE i,350
340    DRAW i,250,1
350 NEXT
360 :
370 FOR i=180 TO 460 STEP 4
380    MOVE i,280
390    DRAW i,180,2
400 NEXT
410 :
420 FOR i=50 TO 200 STEP 4
430    MOVE i,300
440    DRAW i,120,3
450 NEXT
460 :
470 FOR i=100 TO 200 STEP 4
480    MOVE i,380
490    DRAW i,20,0
500 NEXT
510 :
520 FOR i=100 TO 500 STEP 4
530    MOVE i,200
540    DRAW i,50,5
550 NEXT
560 :
570 FOR i=400 TO 600 STEP 4
580    MOVE i,350
590    DRAW i,100,6
600 NEXT
```

Programm 5.45: Löschen von Doppelbelegungen im AND-Modus

Soweit zu den Gestaltungsmöglichkeiten von Grafik. Wir wollen an dieser Stelle erneut betonen, daß der Spaß am Erzeugen von Grafiken in der Nutzung der eigenen Phantasie liegt und nicht nur bei der Verarbeitung von vorgegebenen Programmen. Wir haben versucht, Ihnen aus den vielfältigen Möglichkeiten einige Wege aufzuzeigen.

Kapitel 6

Dreidimensionale Grafiken

Sicher hat sie jeder schon einmal irgendwo bewundert. CAD-Systeme verwenden sie, in Filmen und Videos sieht man sie, und auch Hersteller – vor allem technischer Produkte – werben damit für ihre Waren.

Doch auch selbstgeschriebene Programme können durch 3D-Grafikunterstützung wesentlich komfortabler gestaltet werden. Im folgenden finden Sie eine leicht verständliche Einführung in einige Grundlagen der Darstellung räumlicher Strukturen mit dem Rechner. Dabei werden zwei Arten von 3D-Grafik behandelt: erstens die Darstellung räumlicher geometrischer Körper (Polyeder = Vielecke), zweitens das Zeichnen räumlicher mathematischer Funktionen auf dem Bildschirm.

Doch zunächst zu den Polyedern.

6.1 DARSTELLUNG RÄUMLICHER FIGUREN UND RÄUMLICHE TRANSFORMATIONEN

Polyeder sind räumliche Strukturen, die aus Ecken, Kanten und Flächen (Polygonen) bestehen, wie z. B. Würfel, Pyramiden, Rhomben. Kristalle und Brillanten kann man als besonders schön geformte Polyeder betrachten. Theoretisch läßt sich jeder dreidimensionale Gegenstand angenähert als Polyeder darstellen, wenn man nur genügend viele Eckpunkte bestimmt. Als Beispiel soll uns im folgenden der Würfel von Abb. 6.1 dienen. Er besteht aus 8 Eckpunkten:

Punkt	x	y	z
1	100	100	100
2	−100	100	100
3	−100	−100	100
4	100	−100	100
5	100	100	−100
6	−100	100	−100
7	−100	−100	−100
8	100	−100	−100

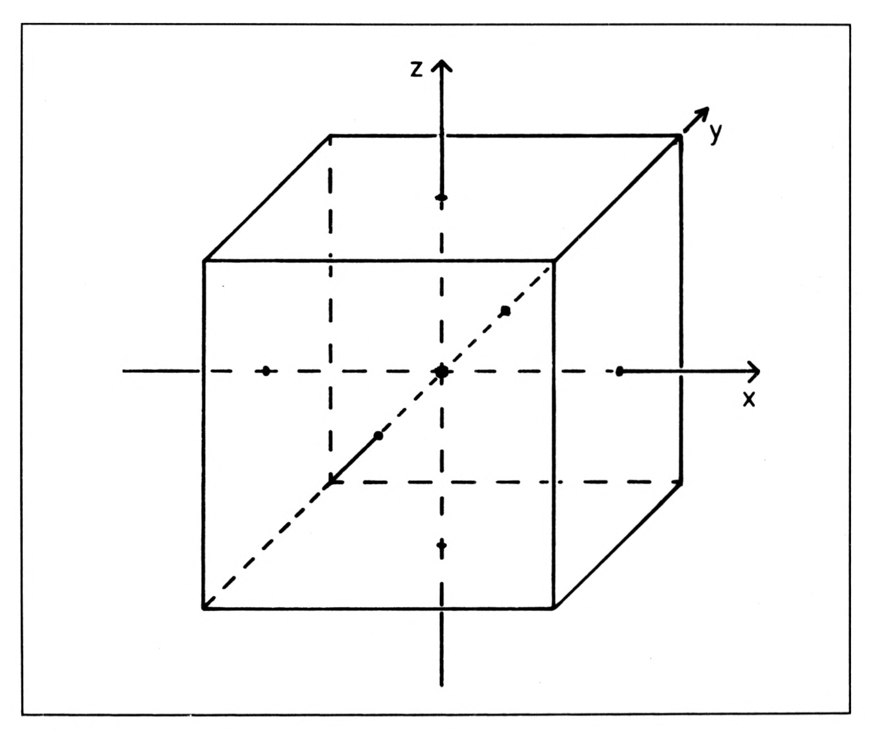


Abb. 6.1: Ein Würfel im Koordinatensystem

. . . und zwölf Linien:

Linie	Anfangspunkt	Endpunkt
1	1	2
2	2	3
3	3	4
4	4	1
5	5	6
6	6	7
7	7	8
8	8	5
9	1	5
10	2	6
11	3	7
12	4	8

Die Eckpunktkoordinaten beziehen sich auf das in Abb. 6.1 eingezeichnete Koordinatensystem. Dieses Koordinatensystem wird im vorliegenden Kapitel beibehalten. Die Ebene, die durch die x- und die z-Achse festgelegt wird, soll parallel zum Bildschirm sein, und die y-Achse soll in den Monitor hineinragen.

Damit man sich den Aufbau dieses Koordinatensystems leicht merken kann, gibt es eine Faustregel (im wahrsten Sinne des Wortes). Halten Sie Ihre rechte (!) Hand vor sich, so daß die offene Handfläche nach oben zeigt, und machen Sie eine Faust. Nun spreizen Sie den Daumen nach rechts ab. Er zeigt jetzt in die Richtung der x-Achse. Dann strecken Sie den Zeigefinger nach vorne; er soll die y-Achse darstellen. Zum Schluß strecken Sie noch den Mittelfinger, so daß er zur Decke zeigt. Er stellt die z-Achse dar.

Das Koordinatensystem ist in Abb. 6.2 noch einmal perspektivisch dargestellt. In den einzelnen Koordinatenebenen, die zur besseren Übersicht alle

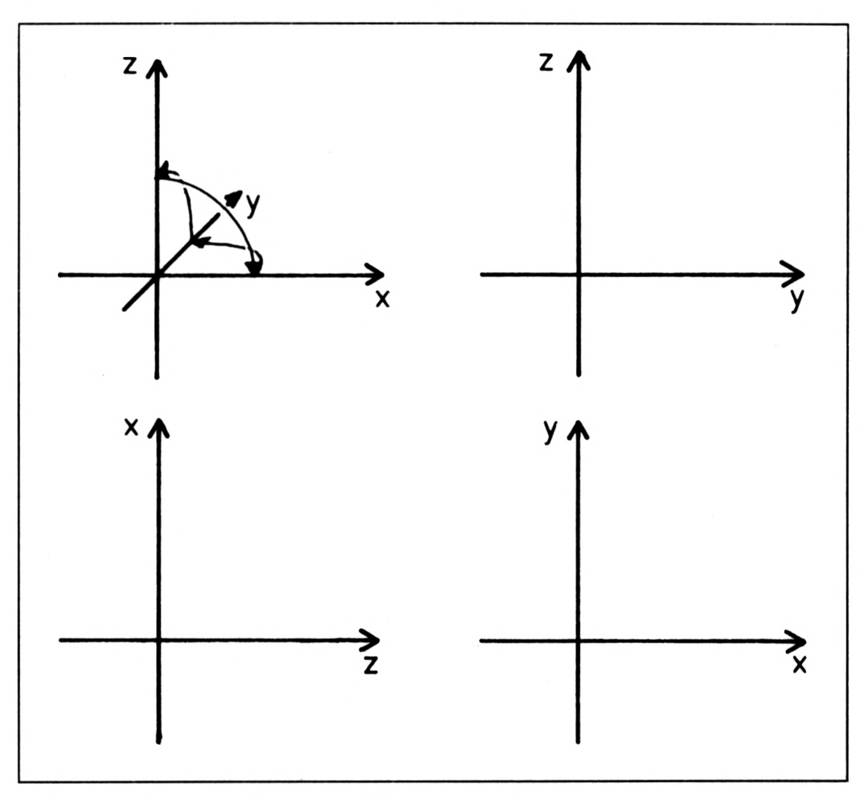


Abb. 6.2: Aufbau des im folgenden benutzten Koordinatensystems

noch einmal einzeln gezeichnet sind, ist der mathematisch positive Drehsinn, auf den wir später noch zu sprechen kommen, mittels gekrümmter Pfeile dargestellt. Die einzelnen Koordinatenebenen sind so gezeichnet, daß die nicht sichtbare dritte Achse zum Betrachter zeigt. Wem räumliche Koordinatensysteme noch nicht so geläufig sind, der kann obiges mit der Merkregel für die rechte Hand nachprüfen.

Damit ist also einheitlich festgelegt, wie wir räumliche Strukturen in den Computer eingeben. Wie machen wir nun solche Strukturen auf dem Bildschirm sichtbar?

Die Grundlage aller Verfahren liegt darin, das räumliche Gebilde so umzuformen, daß es in einer Ebene liegt.

Die einfachste Methode ist, einfach einen Grundriß zu zeichnen, wie Sie ihn sicher von Architekturzeichnungen und Plänen her kennen. Alle Punkte und Linien werden in die Grundfläche (x/y-Ebene) projiziert. Man kann sich vorstellen, der Würfel (als Drahtmodell zur Darstellung der Kanten) würde auf einer ebenen Fläche stehen und von der Sonne (senkrecht von oben) beschienen. Die Schattenlinien stellen den Grundriß dar. Mit dem Rechner läßt sich das bewerkstelligen, indem man beim Zeichnen nur die x- und y-Koordinaten verwendet und die z-Koordinaten unberücksichtigt läßt. Wir verwenden den Bildschirm als x/y-Ebene, und zwar die senkrechten Koordinaten als y-Koordinaten und die waagerechten als x-Koordinaten. Ein kleiner Programmausschnitt soll das verdeutlichen. Zunächst werden die Punkte und Linien in DATA-Zeilen eingegeben, damit das Programm nicht nach jedem Start neu gefüttert werden muß.

```
7040 DATA 8:REM Anzahl der Eckpunkte
7050 DATA 100,100,100,-100,100,100,-100,-100,100,100,-100,100
7060 DATA 100,100,-100,-100,100,-100,-100,-100,-100,100,
          -100,-100

7120 DATA 12:REM Anzahl der Kanten
7130 DATA 1,2,2,3,3,4,4,1
7140 DATA 5,6,6,7,7,8,8,5
7150 DATA 1,5,2,6,3,7,4,8
```

Diese Daten werden in die Variablenfelder x(i), y(i), z(i) für die Koordinaten und ap(i), ep(i) für Anfangspunkte und Endpunkte der Linien eingelesen. Diese Felder müssen natürlich vorher dimensioniert werden:

```
1090 DIM ap(100),ep(100)
1100 DIM x(100),y(100),z(100)
```

```
1360 :
1370 RESTORE
1380 :
1390 READ n:REM Anzahl der Eckpunkte
1400 :
1410 FOR i=1 TO n
1420        READ x(i),y(i),z(i)
1430 NEXT
1440 :
1450 READ k:REM Anzahl der Linien
1460 :
1470 FOR i=1 TO k
1480        READ ap(i),ep(i)
1490 NEXT
```

Jetzt kommen wir zur eigentlichen Zeichen-Routine. Vorher wählen wir jedoch noch den Bildschirmmodus 2 und legen den Koordinatenursprung in die Bildmitte.

```
1050 MODE 2
1060 :
1130 ORIGIN 320,200
1200 :
2000 FOR i=1 TO k
2010        MOVE x(ap(i)),y(ap(i))
2020        DRAW x(ep(i)),y(ep(i))
2030 NEXT
```

Natürlich kann man mit solchen Projektionen den Würfel auch von den Seiten betrachten. Verwendet man statt der y-Koordinaten die z-Koordinaten, so erhält man einen sogenannten Aufriß des Würfels. Durch Verwendung der y-Koordinaten statt der x-Koordinaten und der z- statt der y-Koordinaten ergibt sich ein Seitenriß (Projektion in die y/z-Ebene). Das Ergebnis dieser Projektion dürfte Sie allerdings wenig befriedigen, denn das Ziel dieses Kapitels soll die „räumliche", also die perspektivische, Darstellung sein. Diese Projektionen sind jedoch oft sehr nützlich, wenn man sich von einem beliebigen (nicht bekannten) Gegenstand ein Bild machen will.

Eine unkomplizierte Art der perspektivischen Darstellung läßt sich durch Abwandlung der Zeichen-Routine erreichen. Man zählt einfach bei der Darstellung des Aufrisses zur x- und zur z-Koordinate ein Drittel der y-Koordinate hinzu, wodurch man ein sogenanntes Schrägbild erhält. Die Punkte, die hinter der Bildebene (x/z-Ebene) liegen, werden nach rechts oben verschoben, dieje-

nigen, die vor der Bildebene liegen, nach links unten. Je weiter die Punkte von der Bildebene entfernt sind, desto größer ist die Verschiebung. Das Ergebnis dieses Verfahrens können Sie in Abb. 6.1 betrachten. Der Würfel ist nach diesem Prinzip gezeichnet. Hier die entsprechenden Änderungen:

```
2010 MOVE x(ap(i))+y(ap(i))/3,z(ap(i))+y(ap(i))/3
2020 DRAW x(ep(i))+y(ep(i))/3,z(ep(i))+y(ep(i))/3
```

Wenn Sie die Zeichen-Routine wie oben angegeben ändern, zeichnet das Programm den Würfel so, wie Sie ihn in Abb. 6.1 sehen. Der Bildschirm entspricht dabei der x/z-Ebene.

Bei dieser Art der Darstellung handelt es sich, wie beim Grundriß, um eine Pa-

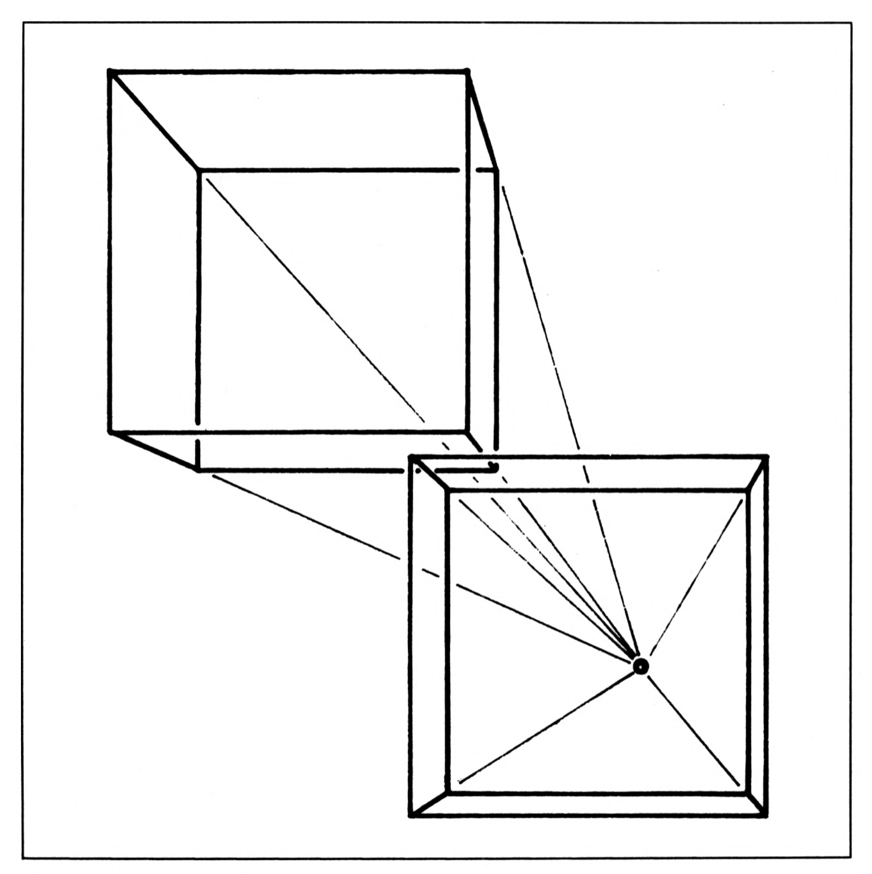


Abb. 6.3: Zentralprojektion

rallelprojektion. Der Würfel wird also so abgebildet, als wäre das Projektionszentrum unendlich weit entfernt. Ähnliche Projektionsbilder erhalten wir, wie schon erwähnt, als Schatten, wenn der Würfel von der Sonne bestrahlt wird. Dabei ist die Sonne das Projektionszentrum (der Punkt, in dem alle Projektionsstrahlen zusammentreffen).

Wenn wir den Würfen direkt anschauen, dann liegt das Projektionszentrum innerhalb unseres Auges, und wir erhalten ein Abbild des Würfels, so wie es in Abb. 6.3 dargestellt ist. Das Abbildungsverfahren, durch das die Zeichnung entstanden ist, nennt man Zentralprojektion.

Bei der Zentralprojektion wählt man eine beliebige Abbildungsebene. Dann verschiebt man einen Punkt des abzubildenden Körpers auf der Projektionsgeraden, die den Punkt mit dem Projektionszentrum verbindet, in die Bildebene. In Abb. 6.4 ist das Prinzip der Zentralprojektion dargestellt. Der Betrachter (dessen Auge das Projektionszentrum ist) schaut in einen nach vorne offenen Würfel hinein. Die Bildebene liegt zwischen dem Würfel und dem Betrachter, wodurch man ein aufrechtstehendes Abbild des Würfels erhält. Würde die Bildebene hinter dem Projektionszentrum liegen, wie es beim Auge der Fall ist, so würde man ein auf dem Kopf stehendes, am Projektionszentrum punktgespiegeltes Abbild bekommen. (Bei Abb. 6.4 handelt es sich ebenfalls um eine Zentralprojektion.)

Jedem Punkt P wird also in der Bildebene ein Punkt P' zugeordnet. Da alle geraden Linien im Raum auf gerade Linien in der Bildebene abgebildet werden,

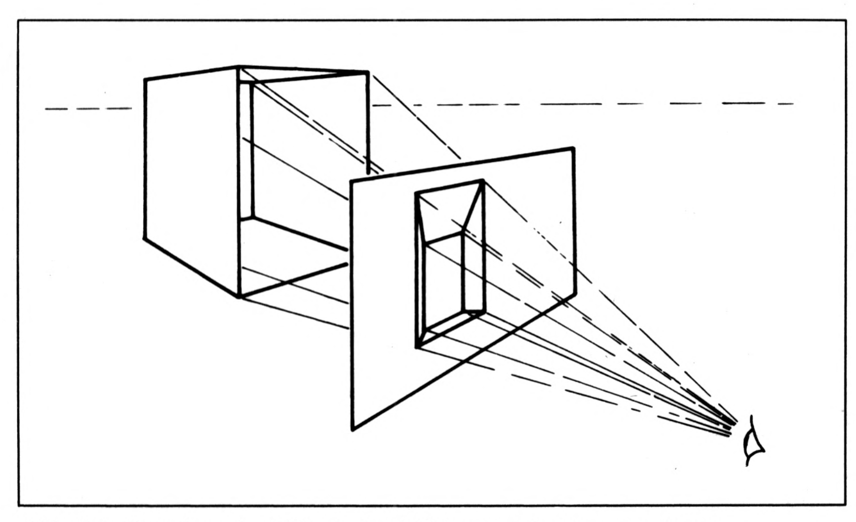


Abb. 6.4: Entstehung des Bildes in der Bildebene bei der Zentralprojektion

kann man einfach die entsprechenden Bildpunkte in der Bildebene miteinander verbinden und erhält so eine vollständig perspektivische Zeichnung des Würfels.

Ein Programm, das eine perspektivische Abbildung erzeugt, braucht also nur die Koordinaten der Bildpunkte zu errechnen und diese dann durch gerade Linien richtig miteinander zu verbinden. Hält man die so entstandene Abbildung so vor sich, daß das Auge sich relativ zur Abbildung an der Stelle befindet, wo man bei der Berechnung der Bildkoordinaten das Projektionszentrum angenommen hat, so werden die Kanten und Ecken des abgebildeten Würfels an dieselben Stellen auf der Augennetzhaut projiziert, als würde der Betrachter den Würfel anschauen. Mit etwas Phantasie entsteht ein räumlicher Eindruck.

Wie erfolgt die Berechnung der Bildkoordinaten? Um die Bildkoordinaten zu berechnen, müssen wir erst einmal einen Beobachtungsstandpunkt und eine Bildebene wählen. Besonders einfach wird die Berechnung, wenn der Beobachtungsstandpunkt (Projektionszentrum) auf einer der Koordinatenachsen liegt und die Bildebene senkrecht auf dieser Achse steht. Eine solche Projektionsanordnung sehen Sie in den Abbildungen 6.5 und 6.6. Das Projektionszentrum Z liegt auf dem negativen Teil der y-Achse, und die Bildebene ist parallel zur x/z-Ebene. Wie man aus der Zeichnung leicht erkennen kann, ist die

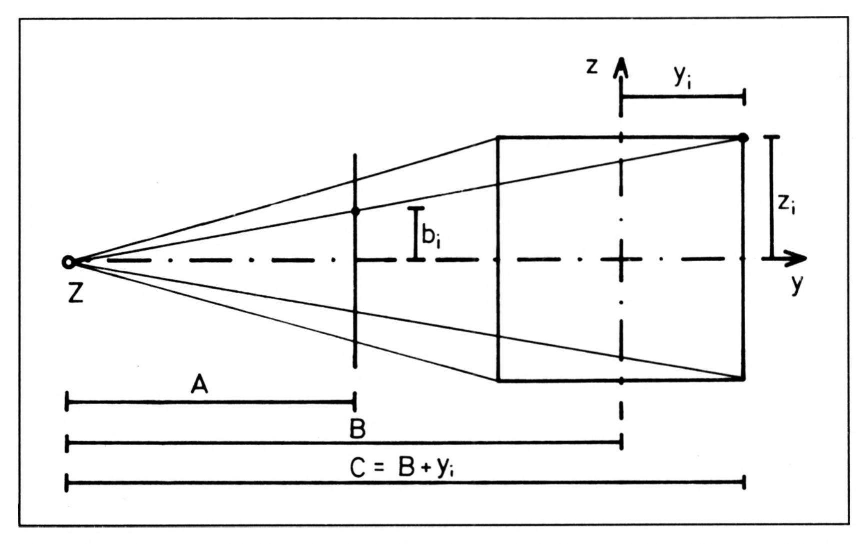


Abb. 6.5: Zentralprojektion. Das Projektionszentrum Z liegt auf dem negativen Teil der y-Achse, und die Bildebene ist parallel zur x/z-Ebene

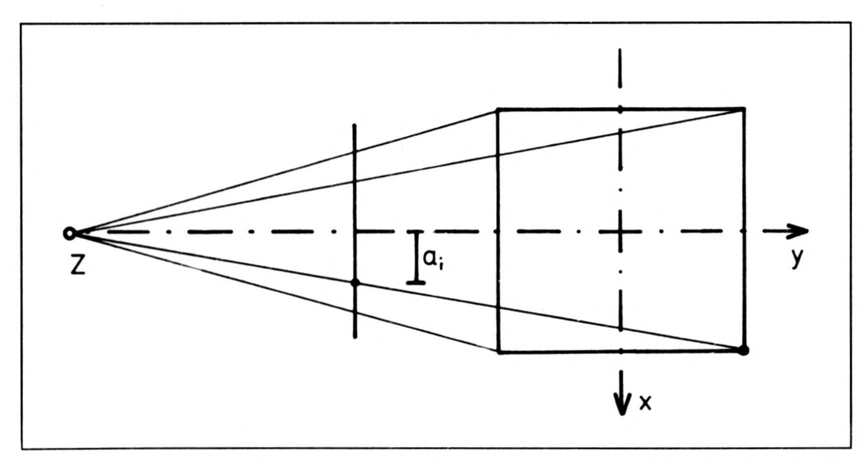


Abb. 6.6: Zentralprojektion. Das Projektionszentrum Z liegt auf dem negativen Teil der y-Achse, und die Bildebene ist parallel zur x/z-Ebene

Größe der Abbildung vom Abstand zwischen Projektionszentrum und Bildebene abhängig. Dieser Abstand ist Ihnen sicher aus der Fotografie schon als Brennweite bekannt.

Die Berechnung der Bildkoordinaten ist jetzt leicht möglich. Man nimmt zunächst eine Ebene, z. B. die y/z-Ebene, und berechnet für einen Punkt des Würfels (oder eines anderen Gegenstandes) die Steigung m des Projektionsstrahls. Diese ergibt sich aus der z-Koordinate des Punktes und der Entfernung vom Projektionszentrum (auf die y-Achse bezogen):

C = B+y(i)
m = z(i)/C
B = Abstand zwischen Koordinatenursprung und Projektionszentrum
C = Horizontalentfernung vom Eckpunkt zum Projektionszentrum

Um die z-Koordinate des Bildpunktes zu erhalten, muß man die Steigung noch mit der Brennweite A multiplizieren.

b(i) = m*A

b(i) gibt die z-Koordinate für die Abbildung an.

Analog verfährt man mit den x-Koordinaten. Die Formeln können Sie sich nach Abb. 6.6 leicht herleiten.

Eine Programmschleife, die aus den Raumkoordinaten die Bildkoordinaten für die Zentralprojektion berechnet, könnte etwa so aussehen:

```
5620 FOR i=1 TO n
5630      c=b+y(i)
5640      x2(i)=x(i)*a/c
5650      z2(i)=z(i)*a/c
5660 NEXT
```

n enthält die Anzahl der Punkte, x(i), y(i), z(i) sind die Raumkoordinaten der Punkte, x2(i), z2(i) die Bildkoordinaten, und a, b, c entsprechen den oben erwähnten Konstanten.

Für das Zeichnen der Linien verwenden wir wieder die Routine für den Grundriß und ändern nur die Variablen ab:

```
5720 FOR i=1 TO k
5730      MOVE x2(ap(i)),z2(ap(i))
5740      DRAW x2(ep(i)),z2(ep(i))
5750 NEXT
```

Für die Darstellung einer Figur stehen uns jetzt genügend Möglichkeiten zur Verfügung.

Richtig interessant wird eine dreidimensionale Grafik aber erst, wenn man die einmal eingegebene Figur verändern und von allen Seiten betrachten kann. Mögliche Veränderungen (man nennt sie auch Transformationen) sind das Vergrößern/Verkleinern, das Spiegeln an einer oder mehreren Achsen und die Drehung.

Verschiebung

Verschiebungen sind sehr einfach zu realisieren. Man zerlegt eine beliebige Verschiebung dazu in drei Teilverschiebungen, die parallel zu den Koordinatenachsen erfolgen. Wollen wir z. B. den Würfel von Abb. 6.1 so verschieben, daß die Ecke mit den Koordinaten (100,100,100) schräg nach (80,120,95) wandert, so addieren wir zu allen x-Koordinaten $80-100=-20$ (Vorzeichen beachten!), zu allen y-Koordinaten $120-100=20$ und zu allen z-Koordinaten $95-100=-5$.

Vergrößern und Verkleinern

Größenveränderungen erhalten wir durch Multiplikation. Nehmen wir an, der Würfel soll in der Höhe auf das Doppelte (zu einem Quader) gestreckt werden. Die Grundfläche soll gleich bleiben. Wir multiplizieren einfach alle z-Koordinaten mit 2. Im Programm wird man aber kaum für jede Achsrichtung eine eigene Schleife zum Vergrößern/Verkleinern schreiben, sondern eine

Schleife für alle drei Achsrichtungen verwenden. Beispiel:

```
6840 FOR i=1 TO n
6850       x(i)=x(i)*mx
6860       y(i)=y(i)*my
6870       z(i)=z(i)*mz
6880 NEXT
```

In so einem Fall ist es nötig, als Faktor für die x- und y-Richtung (mx,my) 1 anzugeben.

Mit der oben angegebenen Schleife kann man nicht nur die Größe der Figur verändern, sondern die Figur auch spiegeln. Der Vergrößerungsfaktor −1 spiegelt die Figur an einer Koordinatenebene. −1 als Faktor für die x-Richtung (mx) bewirkt eine Spiegelung an der y/z-Ebene. Die Faktoren my und mz müssen dabei 1 sein. Wird zusätzlich für my oder mz noch eine −1 gewählt, dann erhalten wir eine räumliche Achsenspiegelung. Sind alle drei Faktoren gleich −1, so hat das eine Punktspiegelung am Koordinatenursprung zur Folge.

In der folgenden Tabelle sind die möglichen Spiegelungen zusammengefaßt:

mx	my	mz	Spiegelung
1	1	1	keine Spiegelung
1	1	−1	an x/y-Ebene
1	−1	1	an z/x-Ebene
1	−1	−1	an x-Achse
−1	1	1	an y/z-Ebene
−1	1	−1	an y-Achse
−1	−1	1	an z-Achse
−1	−1	−1	Punktspiegelung am Ursprung

Drehung

Die nächste Transformationsart, die Drehung, ist etwas komplizierter. Zur Herleitung der zur Drehung nötigen Formeln betrachten wir Abb. 6.7. Die Zeichnung beschreibt die Drehung des Punktes P in der z/y-Ebene um den Winkel α. Durch Drehungen dieser Art kann man die räumliche Figur jeweils um eine Achse drehen. Im dargestellten Fall ist es die z-Achse. Durch Kombinieren von Drehungen um verschiedene Achsen kann man jede räumliche Verdrehung der Figur erreichen. An dieser Stelle kommen wir noch einmal auf den schon erwähnten mathematischen Drehsinn zurück. In der Zeichnung

wird im positiven Sinn (gegen den Uhrzeigersinn) gedreht; d. h. wenn der Drehwinkel positiv ist, wird gegen den Uhrzeigersinn gedreht, ist er negativ, wird mit dem Uhrzeigersinn gedreht.

Um festzustellen, wie in einer bestimmten Koordinatenebene positiv und wie negativ gedreht wird, muß man die Lage der Koordinatenachsen kennen. Wenn Sie die Merkregel für die rechte Hand auf das Koordinatensystem in Abb. 6.7 anwenden, werden Sie feststellen, daß die z-Achse zum Betrachter zeigt. Für den Drehsinn kann man wieder eine Merkregel für die rechte (!) Hand ableiten. Um den positiven Drehsinn um eine Koordinatenachse zu bestimmen, macht man eine Faust und spreizt den Daumen ab. Dann dreht man die Hand so, daß der Daumen in Richtung einer Koordinatenachse zeigt. Die restlichen Finger zeigen jetzt den mathematisch positiven Drehsinn an. Wenn Sie keinen Fehler gemacht haben, werden Sie festgestellt haben, daß in Abb. 6.7 positiv gedreht wird (siehe Pfeilspitze beim Winkel).

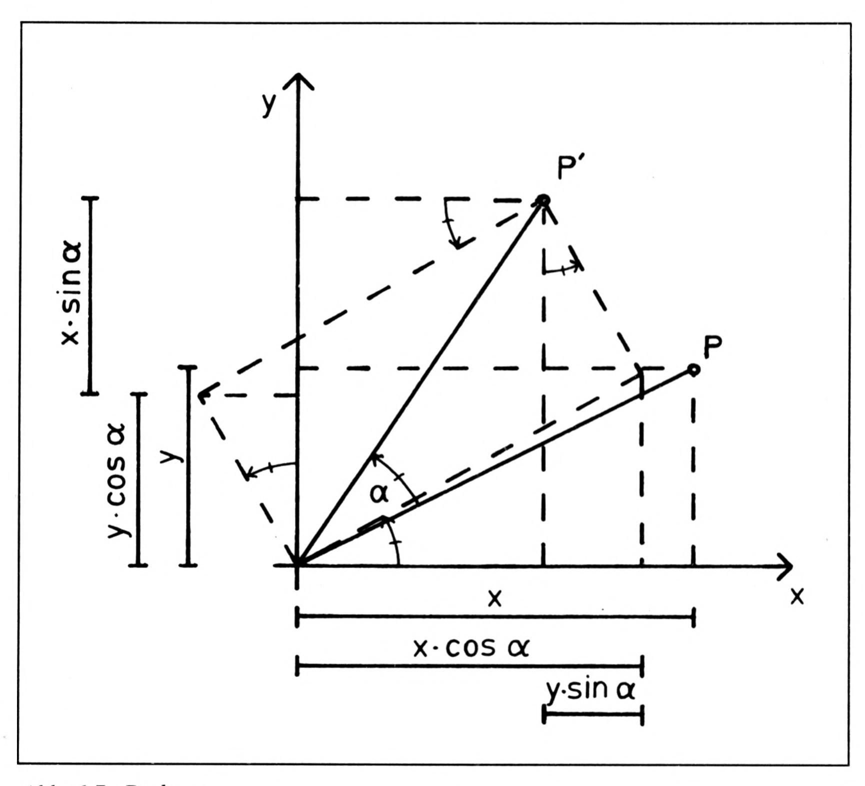


Abb. 6.7: Drehung

Durch Drehen des Punktes P mit den Koordinaten (x,y) um den Winkel α erhalten wir also den neuen Punkt P', der die Koordinaten (x',y') haben soll. Die Koordinatenwerte x und y bilden ein Rechteck, das in Abb. 6.7 gestrichelt eingezeichnet ist.

Zur Orientierung: In einer Ecke des Rechtecks liegt der Koordinatenursprung, diagonal gegenüber der Punkt P.

Bei der Drehung des Punktes P wollen wir in Gedanken das gestrichelte Rechteck mitdrehen. An dem gedrehten Rechteck, das in der Abbildung mit eingezeichnet ist, kann der Leser, der schon mit Winkelfunktionen zu tun gehabt hat, leicht die Formeln für die Bestimmung der Koordinaten von P' ablesen:

$$x' = x * \cos(\alpha) - y * \sin(\alpha)$$
$$y' = x * \sin(\alpha) + y * \cos(\alpha)$$

Diese Formeln lassen sich durch Austauschen der Variablen für die verschiedenen Koordinatenebenen abändern. Dafür stellen wir hier eine Tabelle sich entsprechender Koordinatenvariablen zusammen (siehe auch Abb. 6.2).

Drehung um z-Achse:	x	y
Drehung um y-Achse:	z	x
Drehung um x-Achse:	y	z

Wenn wir die Variablen in den Formeln entsprechend der Tabelle vertauschen, erhalten wir die Formeln für die Drehung um die übrigen Achsen, so daß der richtige Drehsinn beibehalten wird:

Drehung um die y-Achse:

$$z' = z * \cos(\alpha) - x * \sin(\alpha)$$
$$x' = z * \sin(\alpha) - x * \cos(\alpha)$$

Drehung um die x-Achse:

$$y' = y * \cos(\alpha) - z * \sin(\alpha)$$
$$z' = y * \sin(\alpha) + z * \cos(\alpha)$$

Das sind alle Formeln, die man braucht, um eine Figur in jede beliebige Lage drehen zu können. Allerdings ist dabei zu beachten, daß es bei einer kombinierten Drehung auf eine Reihenfolge der verwendeten Drehachsen ankommt. Wenn man erst um α um die z-Achse dreht und dann um β um die y-

Achse, ist das Ergebnis anders, als wenn man erst um β um die y-Achse dreht und dann um α um die z-Achse. Will man die zuerst erwähnte Drehung rückgängig machen, muß man erst um die y-Achse um β zurückdrehen und dann um die z-Achse um α. Dreht man zuerst um die z-Achse zurück und dann um die y-Achse, bekommt man die Figur nicht in die Ausgangsstellung zurück.

Mit der Möglickeit, die Figur beliebig zu drehen, erübrigt sich auch ein Standortwechsel bei der Projektion, und es kann, vor allem bei der Zentralprojektion, der für die Berechnung günstige Standpunkt beibehalten werden. Strukturen, die weit vom Ursprung entfernt sind, sollten vor der Drehung zum Ursprung hin verschoben werden, da sie sonst leicht vom Bildschirm verschwinden können und dann vielleicht nicht mehr zu finden sind.

Unterprogrammsammlung für 3D-Grafik

Im folgenden finden Sie ein Programm, in dem alle bisher besprochenen Transformationen und Projektionen zusammengefaßt sind. Das Programm ist so aufgebaut, daß der Leser einzelne Unterprogramme in eigene Software übernehmen kann. Das Programm enthält auch eine Hardcopy-Routine in Maschinensprache, weshalb es angebracht ist, das Programm vor einem Testlauf zu sichern.

```
1000 REM -----------------------------
1010 REM ---        3-D Grafik      ---
1020 REM --- Unterprogrammsammlung ---
1030 REM -----------------------------
1040 :
1050 MODE 2
1060 :
1070 MEMORY &9FFF
1080 :
1090 DIM ap(100),ep(100)
1100 DIM x1(100),y1(100),z1(100)
1110 DIM x2(100),z2(100)
1120 :
1130 ORIGIN 420,200
1140 WINDOW #0,26,80,1,25
1150 WINDOW #1,1,25,1,25
1160 INK 0,24:INK 1,4
1170 PEN #1,0:PAPER #1,1
1180 PEN #0,1:PAPER #0,0
1190 BORDER 26
1200 :
1210 DEG
1220 :
1230 REM -----------------------------
1240 REM ---  Datas f. Hardcopy    ---
1250 REM -----------------------------
```

Programm 6.1: 3D-Grafik

```
1260 :
1270 RESTORE 8040
1280 :
1290 FOR n=41088 TO 41313
1300     READ byte$
1310     byte=VAL("&"+byte$)
1320     POKE n,byte
1330 NEXT n
1340 :
1350 REM -----------------------------
1360 REM ---  Figurdaten einlesen  ---
1370 REM -----------------------------
1380 :
1390 RESTORE
1400 :
1410 READ n :REM  Anzahl der Eckpunkte
1420 :
1430 FOR i=1 TO n
1440    READ x1(i),y1(i),z1(i):REM Koord.
1450 NEXT
1460 :
1470 READ k :REM     Anzahl der Linien
1480 :
1490 FOR i=1 TO k
1500    READ ap(i),ep(i):REM Anf. Ende
1510 NEXT
1520 :
1530 abb=3
1540 :
2000 REM -----------------------------
2010 REM ---                      ---
2020 REM ---      M E N U E       ---
2030 REM ---                      ---
2040 REM -----------------------------
2050 :
2060 GOSUB 5040
2070 :
2080 CLS #1
2090 PRINT #1," 1 = Drehen"
2100 PRINT #1," 2 = Verschieben"
2110 PRINT #1," 3 = Masstab"
2120 PRINT #1," 4 = Grundriss"
2130 PRINT #1," 5 = Aufriss"
2140 PRINT #1," 6 = Schraegbild"
2150 PRINT #1," 7 = Zentralprojektion"
2160 PRINT #1," 8 = Figur aendern"
2170 PRINT #1," 9 = Grundfigur laden"
2180 PRINT #1," 0 = Hardcopy"
2190 :
2200 a$=INKEY$
2210 :
2220 IF a$<"0" OR a$>"9" THEN 2200
2230 :
2240 IF a$="0" THEN PAPER #1,0:CLS#1:CALL &A080:PAPER #1
,1
2250 IF a$="1" THEN GOTO 2390
2260 IF a$="2" THEN GOTO 2610
```

Programm 6.1: 3D-Grafik (Forts.)

```
2270 IF a$="3" THEN GOTO 2730
2280 IF a$="4" THEN abb=1
2290 IF a$="5" THEN abb=2
2300 IF a$="6" THEN abb=3
2310 IF a$="7" THEN abb=4:w=1
2320 IF a$="8" THEN GOTO 2950
2330 IF a$="9" THEN GOTO 1390
2340 :
2350 ON abb GOSUB 4140,4540,5040,5540
2360 :
2370 GOTO 2080
2380 :
2390 REM ----------------------------
2400 REM ---  Drehung ausfuehren   ---
2410 REM ----------------------------
2420 :
2430 CLS #1
2440 PRINT #1,"Winkel in Grad eingeben:"
2450 PRINT #1,"Drehung um Z-Achse ":INPUT #1,alpha
2460 PRINT #1,"Drehung um Y-Achse ":INPUT #1,beta
2470 PRINT #1,"Drehung um X-Achse ":INPUT #1,gamma
2480 :
2490 GOSUB 6040 :REM      Drehung um Z
2500 GOSUB 6240 :REM      Drehung um Y
2510 GOSUB 6440 :REM      Drehung um X
2520 :
2530 ON abb GOSUB 4140,4540,5040,5540
2540 :
2550 GOTO 2080
2560 :
2570 REM ----------------------------
2580 REM ---Verschiebung ausfuehren---
2590 REM ----------------------------
2600 :
2610 CLS #1
2620 PRINT #1,"Verschiebung:"
2630 PRINT #1,"Verschiebung, X-Achse":INPUT #1,dx
2640 PRINT #1,"Verschiebung, Y-Achse":INPUT #1,dy
2650 PRINT #1,"Verschiebung, Z-Achse":INPUT #1,dz
2660 :
2670 GOSUB 6640 :REM      Verschiebung
2680 :
2690 ON abb GOSUB 4140,4540,5040,5540
2700 :
2710 GOTO 2080
2720 :
2730 REM ----------------------------
2740 REM ---    Masstab aendern    ---
2750 REM ----------------------------
2760 :
2770 CLS #1
2780 PRINT #1,"Masstab aendern:"
2790 PRINT #1,"Faktor f. X-Koord. ":INPUT #1,mx
2800 PRINT #1,"Faktor f. Y-Koord. ":INPUT #1,my
2810 PRINT #1,"Faktor f. z-Koord. ":INPUT #1,mz
2820 :
2830 GOSUB 6800 :REM            Masstab
```

Programm 6.1: 3D-Grafik (Forts.)

```
2840 :
2850 ON abb GOSUB 4140,4540,5040,5540
2860 :
2870 GOTO 2080
2880 :
2890 REM ----------------------------
2900 REM ---    Figur aendern     ---
2910 REM ----------------------------
2920 :
2930 REM ---  Punkte aendern/eing. ---
2940 :
2950 CLS #1
2960 PRINT #1,"Figur aendern"
2970 PRINT #1,"Punkte Nr. X Y Z"
2980 :
2990 FOR i=1 TO n
3000    PRINT #1,i;x1(i);y1(i);z1(i)
3010 NEXT
3020 :
3030 INPUT #1,"Nummer (0=Keine Aenderung)";nr
3040 IF nr=0 THEN 3100
3050 INPUT #1,"X";x1(nr)
3060 INPUT #1,"Y";y1(nr)
3070 INPUT #1,"Z";z1(nr)
3080 GOTO 2950
3090 :
3100 n=n+1
3110 PRINT #1,"neuer Punkt (Keine Eing.=Ende)"
3120 INPUT #1,"X",x$
3130 INPUT #1,"Y";y$
3140 INPUT #1,"Z";z$
3150 IF x$="" THEN n=n-1:GOTO 3210
3160 x1(n)=VAL(x$):y1(n)=VAL(y$):z1(n)=VAL(z$)
3170 GOTO 3100
3180 :
3190 REM ---  Linien aendern/eing. ---
3200 :
3210 CLS #1
3220 PRINT #1,"Linien  Nr. AP  EP"
3230 :
3240 FOR i=1 TO k
3250    PRINT #1,i;ap(i);ep(i)
3260 NEXT
3270 :
3280 INPUT #1,"Nummer (0=Keine Aenderung)";nr
3290 IF nr=0 THEN 3340
3300 INPUT #1,"AP";ap(nr)
3310 INPUT #1,"EP";ep(nr)
3320 GOTO 3210
3330 :
3340 k=k+1
3350 PRINT #1,"neue Linie (Keine Eing.=Ende)"
3360 INPUT #1,"AP",ap$
3370 INPUT #1,"EP";ep$
3380 IF ap$="" THEN k=k-1:GOTO 2060
3390 ap(k)=VAL(ap$):ep(k)=VAL(ep$)
3400 GOTO 3340
```

Programm 6.1: 3D-Grafik (Forts.)

```
3410 :
3420 *********************************
3430 :
4000 REM -----------------------------
4010 REM ---                       ---
4020 REM ---    Unterprogramme     ---
4030 REM ---                       ---
4040 REM -----------------------------
4050 :
4100 REM -----------------------------
4110 REM ---      Grundriss        ---
4120 REM -----------------------------
4130    :
4140    CLS
4150    :
4160    FOR i=1 TO k
4170       MOVE x1(ap(i)),y1(ap(i))
4180       DRAW x1(ep(i)),y1(ep(i))
4190    NEXT
4200    :
4210 RETURN
4220 :
4500 REM -----------------------------
4510 REM ---       Aufriss         ---
4520 REM -----------------------------
4530    :
4540    CLS
4550    :
4560    FOR i=1 TO k
4570       MOVE x1(ap(i)),z1(ap(i))
4580       DRAW x1(ep(i)),z1(ep(i))
4590    NEXT
4600    :
4610 RETURN
4620 :
5000 REM -----------------------------
5010 REM ---     Schraegbild       ---
5020 REM -----------------------------
5030    :
5040    CLS
5050    :
5060    FOR i=1 TO k
5070       MOVE x1(ap(i))+y1(ap(i))/3,z1(ap(i))+y1(ap(i)
)/3
5080       DRAW x1(ep(i))+y1(ep(i))/3,z1(ep(i))+y1(ep(i)
)/3
5090    NEXT
5100    :
5110 RETURN
5120 :
5500 REM -----------------------------
5510 REM ---   Zentralprojektion   ---
5520 REM -----------------------------
5530    :
5540    IF w=0 THEN 5620
5550    CLS #1
5560    PRINT #1,"Beobachtungsstandort ":INPUT #1,b
```

Programm 6.1: 3D-Grafik (Forts.)

```
5570     PRINT #1,"Brennweite ":INPUT #1,a
5580     w=0
5590     :
5600 REM --- Projektion berechnen  ---
5610     :
5620     FOR i=1 TO n
5630        c=-b+y1(i)
5640        z2(i)=z1(i)*a/c
5650        x2(i)=x1(i)*a/c
5660     NEXT
5670     :
5680 REM ---       Darstellen        ---
5690     :
5700     CLS
5710     :
5720     FOR i=1 TO k
5730        MOVE x2(ap(i)),z2(ap(i))
5740        DRAW x2(ep(i)),z2(ep(i))
5750     NEXT
5760     :
5770 RETURN
5780 :
6000 REM -----------------------------
6010 REM ---       Drehung um Z      ---
6020 REM -----------------------------
6030     :
6040     FOR i=1 TO n
6050        x=x1(i)
6060        y=y1(i)
6070        x1(i)=x*COS(alpha)-y*SIN(alpha)
6080        y1(i)=x*SIN(alpha)+y*COS(alpha)
6090     NEXT
6100     :
6110 RETURN
6120 :
6200 REM -----------------------------
6210 REM ---       Drehung um Y      ---
6220 REM -----------------------------
6230     :
6240     FOR i=1 TO n
6250        z=z1(i)
6260        x=x1(i)
6270        z1(i)=z*COS(beta)-x*SIN(beta)
6280        x1(i)=z*SIN(beta)+x*COS(beta)
6290     NEXT
6300     :
6310 RETURN
6320 :
6400 REM -----------------------------
6410 REM ---       Drehung um X      ---
6420 REM -----------------------------
6430     :
6440     FOR i=1 TO n
6450        y=y1(i)
6460        z=z1(i)
6470        y1(i)=y*COS(gamma)-z*SIN(gamma)
6480        z1(i)=y*SIN(gamma)+z*COS(gamma)
```

Programm 6.1: 3D-Grafik (Forts.)

```
6490    NEXT
6500    :
6510 RETURN
6520 :
6600 REM -----------------------------
6610 REM ---      Verschiebung      ---
6620 REM -----------------------------
6630    :
6640    FOR i=1 TO n
6650       x1(i)=x1(i)+dx
6660       y1(i)=y1(i)+dy
6670       z1(i)=z1(i)+dz
6680    NEXT
6690    :
6700 RETURN
6710 :
6800 REM -----------------------------
6810 REM ---        Masstab         ---
6820 REM -----------------------------
6830    :
6840    FOR i=1 TO n
6850       x1(i)=x1(i)*mx
6860       y1(i)=y1(i)*my
6870       z1(i)=z1(i)*mz
6880    NEXT
6890    :
6900 RETURN
6910 :
6920 *********************************
6930 :
7000 REM -----------------------------
7010 REM --- Eckpunkts-Koordinaten ---
7020 REM -----------------------------
7030 :
7040 DATA 19
7050 DATA 30,0,0,30,-60,0,-30,-60,0,-30,0,0,-70,0,0,-70,
180,0,70,180,0,70,0,0
7060 DATA 30,0,100,30,-60,100,-30,-60,100,-30,0,100,-70,
0,50,-70,180,50,70,180,50,70,0,50
7070 DATA 0,-30,170,0,0,90,0,180,90
7080 :
7090 REM -----------------------------
7100 REM ---   Daten fuer Linien   ---
7110 REM -----------------------------
7120 :
7130 DATA 32
7140 DATA 1,2,2,3,3,4,5,6,6,7,7,8,8,5,1,9,2,10,3,11,4,12
,5,13,6,14,7,15,8,16
7150 DATA 9,10,10,11,11,12,13,14,14,15,15,16,16,13
7160 DATA 11,17,12,17,9,17,10,17,13,18,14,19,15,19,16,18
,18,19,9,12
7170 :
7180 *********************************
7190 :
8000 REM -----------------------------
8010 REM ---  Daten fuer Hardcopy  ---
8020 REM -----------------------------
```

Programm 6.1: 3D-Grafik (Forts.)

```
8030 :
8040 DATA cd,c6,bb,e5,d5,cd,cc,bb,e5,d5,cd,d5,bb,e5,d5,c
d,d8,bb,e5,d5
8050 DATA cd,e1,bb,47,cd,e7,bb,4f,c5,ed,73,60,a1,cd,ba,b
b,3e,1b
8060 DATA cd,13,a1,3e,31,cd,13,a1,21,8f,01,cd,47,a1,3e,1
b,cd,13,a1
8070 DATA 3e,4c,cd,13,a1,3e,00,cd,13,a1,3e,03,cd,13,a1,0
6,80,3e,00
8080 DATA cd,13,a1,10,f9,11,00,00,e5,06,07,0e,00,e5,cd,5
2,a1,cb,11
8090 DATA e1,2b,2b,10,f5,79,e1,e5,01,08,00,ed,42,30,02,e
6,78
8100 DATA cd,13,a1,13,21,81,02,ed,52,e1,20,d8,b7,01,0e,0
0
8110 DATA ed,42,30,ad,3e,1b,cd,13,a1,3e,32,cd,13,a1,cd,4
7,a1,18,08
8120 DATA 4f,cd,1b,bb,fe,fc,20,22,ed,7b,60,a1,c1,79,cd,e
4,bb
8130 DATA 78,cd,de,bb,d1,e1,cd,d2,bb,d1,e1,cd,cf,bb,d1,e
1,cd,c9,bb
8140 DATA d1,e1,cd,c0,bb,c9,cd,2e,bd,38,d2,79,cd,2b,bd,c
9
8150 DATA 3e,0d,cd,13,a1,3e,0a,cd,13,a1,c9
8160 DATA c5,d5,e5,cd,f0,bb,b7,28,01,37,e1,d1,c1,c9,00,0
0
```

Programm 6.1: 3D-Grafik (Forts.)

Zunächst die Variablenübersicht:

x1(i), y1(i), z1(i):	Raumkoordinaten
x2(i), z2(i):	Bildkoordinaten für Zentralprojektion
ap(i), ep(i):	Anfangs- und Endpunkt der Linie i
abb:	Merkvariable für Abbildungsart
a$:	Hilfsvariable für Tastaturabfrage
alpha, beta, gamma:	Winkel für Drehung um z-, y- und x-Achse
dx, dy, dz:	Beträge der Verschiebungen in x-, y- und z-Richtung
mx, my, mz:	Faktoren für Größenveränderungen in x-, y- und z-Richtung
n:	Anzahl der Eckpunkte
k:	Anzahl der Verbindungslinien (Kanten)
i:	Laufvariable für Schleifen
x$, y$, z$:	Hilfsvariablen für Eingabe neuer Punkte
ap$, ep$:	Hilfsvariablen für Eingabe neuer Linien
nr:	Hilfsvariable (Zeiger) für Korrektur von Linien und Punkten
b:	Beobachtungsstandort für Zentralprojektion
a:	Brennweite

c: Hilfsvariable
x, y, z: Hilfsvariablen für Koordinatendrehung
w: Hilfsvariable. Wenn w=1 ist, stehen keine Parameter für die Zentralprojektion (a, b) zur Verfügung, und die Parameter müssen erst abgefragt werden.

Programmbeschreibung

Zeile 1050: Bildschirmmodus 2 (640*200) Punkte.

Zeile 1070: Verschieben des Endes des BASIC-RAM, um für die Hardcopy-Routine Platz zu schaffen. Sollten Sie auf die Hardcopy-Routine verzichten wollen, entfällt diese Zeile. Außerdem entfallen dann die Zeilen 1210 bis 1320, 2180, 2240 und 8000 bis 8160.

Zeilen 1090 bis 1110: Dimensionierung der Variablenfelder.

Zeilen 1130 bis 1190: Der Koordinatenursprung für die Grafik wird nach (420/200) verschoben. Das Grafikfenster (#0) wird festgelegt, um die Grafik gezielt löschen zu können. Über Window #1 erfolgt der Dialog mit dem Benutzer. Weiter werden die Farben und Farbkombinationen für die Fenster und die Randfarbe festgelegt.

Zeile 1210: Umschalten der Winkelfunktionen auf Gradmaß (360 Grad).

Zeilen 1230 bis 1340: Einleseschleife für Hardcopy-Routine. Um Programmabstürzen vorzubeugen, sollte ein Probelauf mit zwei geänderten Zeilen erfolgen:

```
1300 summe =summe+byte
1320 IF summe<>&76EF THEN PRINT
     "Fehler":STOP
```

Zeilen 1350 bis 1520: Einlesen der Figurdaten. In Zeile 1390 wird der Zeiger für die Daten auf den Anfang gesetzt. In Zeile 1410 wird die Anzahl der Eckpunkte eingelesen, damit in der folgenden Schleife die richtige Anzahl Koordinaten gelesen wird. Das Einlesen der Anfangs- und Endpunkte für die Linien erfolgt nach dem gleichen Prinzip.

Zeile 1530: Voreinstellung der Projektionsart.

Zeilen 2000 ff.: Beginn des Haupt-(oder Steuer-)programms.

Zeile 2060: Schrägbild der Figur zeichnen.

Zeilen 2080 bis 2180: Fenster #1 löschen und Menü anzeigen.

Zeile 2200:	Tastaturabfrage.
Zeile 2220:	Prüfen auf (richtige) Eingabe.
Zeilen 2240 bis 2330:	Eingabe auswerten. Für das Erstellen einer Hardcopy (Zeile 2240) genügt ein Befehl, jedoch wird dazu erst das Eingabefenster gelöscht. In den Zeilen 2280 bis 2310 wird festgelegt, welche Projektionsart verwendet wird.
Zeile 2350:	Nach jeder Eingabe wird zu dem Unterprogramm verzweigt, das die Figur in der ausgewählten Projektionsart auf den Bildschirm bringt.
Zeile 2370:	Rücksprung zur Menüanzeige und Tastaturabfrage.
Zeilen 2390 bis 2560:	Abfrage der Drehwinkel und Sprung zu den Unterprogrammen. Nach Ausführung der Drehung wird zum vorgewählten Projektionsprogramm gesprungen.
Zeilen 2570 bis 2720:	Das Ausführen der Verschiebung erfolgt analog zur Drehung.
Zeilen 2730 bis 2880:	Wie oben.
Zeilen 2890 ff.:	Hier kann die Figur abgeändert oder um Punkte oder Linien ergänzt werden. Punkte kann man „löschen", indem man z. B. ihre Koordinaten so ändert, daß sie mit einem anderen Punkt zusammenfallen. Beim Löschen von Linien setzt man einfach den Endpunkt EP gleich dem Anfangspunkt AP.
Zeilen 2990 bis 3010:	Anzeige der vorhandenen Punkte.
Zeilen 3030 bis 3080:	Nach Eingabe einer Punktnummer fragt das Programm nach neuen Koordinaten für diesen Punkt. Wenn man als Punktnummer 0 eingibt, verzweigt das Programm weiter.
Zeilen 3100 bis 3170:	Hier können zusätzlich Punkte eingegeben werden. Möchten Sie keine weiteren Punkte eingeben, dann drücken Sie dreimal ENTER.
Zeilen 3240 bis 3260:	Anzeige der vorhandenen Linien.
Zeilen 3280 bis 3320:	Hier können Sie Linien ändern. Abbruch durch Eingabe von 0.
Zeilen 3340 bis 3400:	Die Figur kann durch zusätzliche Linien ergänzt werden. Wird keine Eingabe gemacht (nur ENTER drücken), kommt man wieder zum Menü.

Zeilen 4000 ff.:	Unterprogrammabschnitt. Hier beginnt der eigentliche Grafikteil.
Zeilen 4100 bis 4210:	Der Grundriß der Figur wird gezeichnet. CLS in Zeile 4140 löscht das Grafikfenster.
Zeilen 4500 bis 4610:	Aufriß zeichnen. Die Figur wird von der negativen y-Achse her betrachtet. Unterscheidet sich vom vorhergehenden Programm nur in den verwendeten Variablen.
Zeilen 5000 bis 5110:	Zeichnen eines Schrägrisses, wie vorher beschrieben.
Zeilen 5550 bis 5770:	Berechnen und Zeichnen der Zentralprojektion. Die Zeilen 5550 bis 5580 werden nur durchlaufen, wenn vorher im Menü die Taste 7 gedrückt wurde. Das Programm fragt dann nach dem Betrachtungsstandort (auf der y-Achse) und der Brennweite. Wenn Sie das Objekt, wie in Abb. 6.5 gezeigt, betrachten wollen, müssen Sie eine negative Zahl als Standort eingeben, da dann der Standort auf dem negativen Teil der y-Achse liegt. In den Zeilen 5620 bis 5660 erfolgt die Berechnung der Bildkoordinaten, wie schon beschrieben, und in den Zeilen 5720 bis 5750 wird die Figur nach dem gleichen Verfahren wie beim Grundriß gezeichnet.
Zeilen 6000 bis 6120:	Drehung um die z-Achse. Die z-Koordinaten bleiben dabei erhalten. Die x- und y-Koordinaten müssen in den Variablen x und y zwischengespeichert werden.
Zeilen 6200 bis 6520:	Drehungen um y- und x-Achse.
Zeilen 6600 bis 6700:	Verschiebung der Figur.
Zeilen 6800 bis 6900:	Größe verändern bzw. spiegeln.
Zeilen 7000 bis 7070:	Daten für Eckpunkte. Das erste Element gibt die Anzahl der Eckpunkte an. Die restlichen Zahlen geben, in Dreiergruppen, die x-, y- und z-Koordinaten an.
Zeilen 7990 bis 7160:	Daten für Linien. Das erste Element gibt die Anzahl der Linien an. Die weiteren Zahlen gehören paarweise zusammen und geben die Nummern von Anfangs- und Endpunkt der Linien an.
Zeilen 8000 bis 8160:	Hex-Codes für Hardcopy.

Zum Abschluß noch ein paar Darstellungen der mit obigem Programm projizierten Figur (siehe Abb. 6.8).

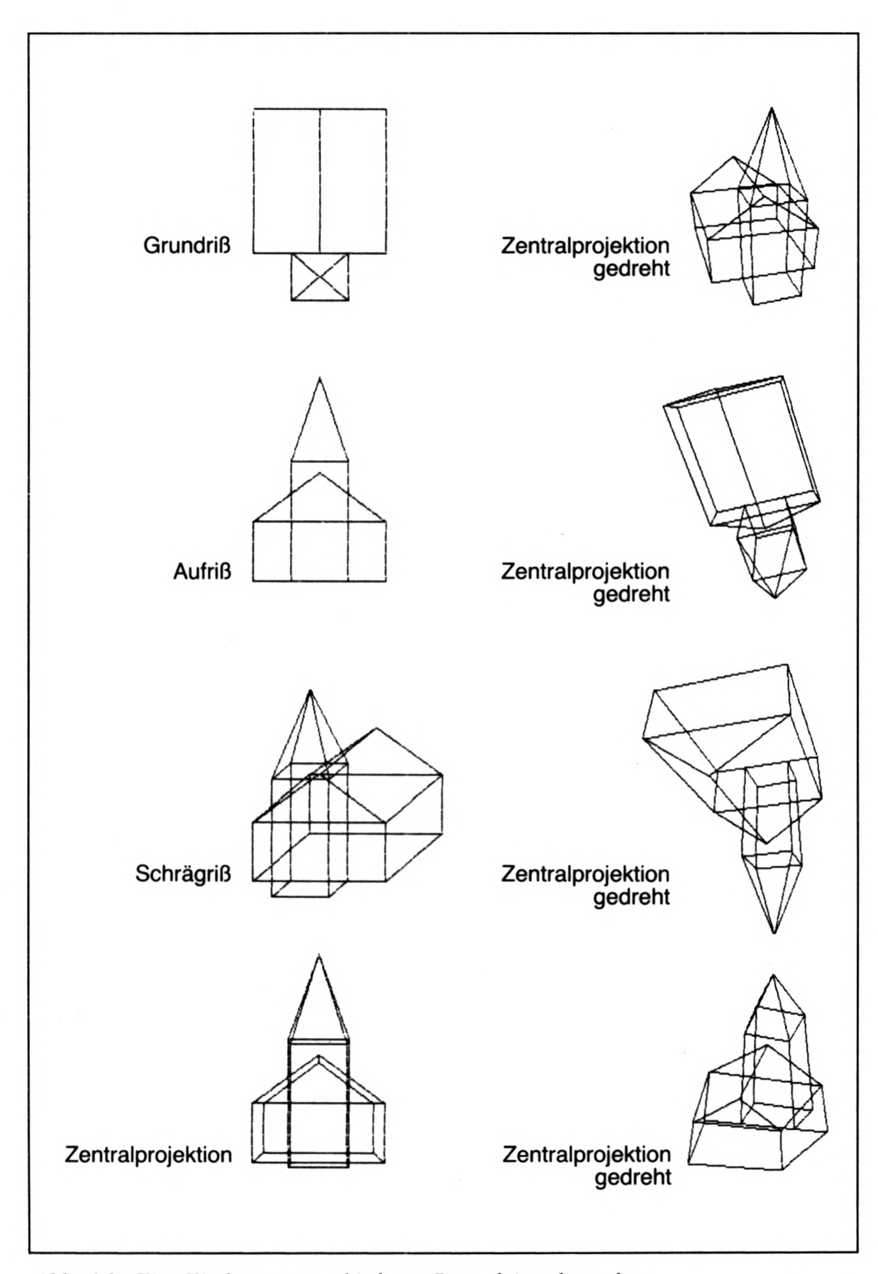


Abb. 6.8: Eine Kirche, aus verschiedenen Perspektiven betrachtet

Hier noch ein anderes Beispiel (Abb. 6.9). Die Einleseschleife für die Figurdaten wurde hier ersetzt durch zwei verschachtelte Schleifen, die mit geeigneten Formeln Punkte ermitteln und diese miteinander verbinden. Vorausset-

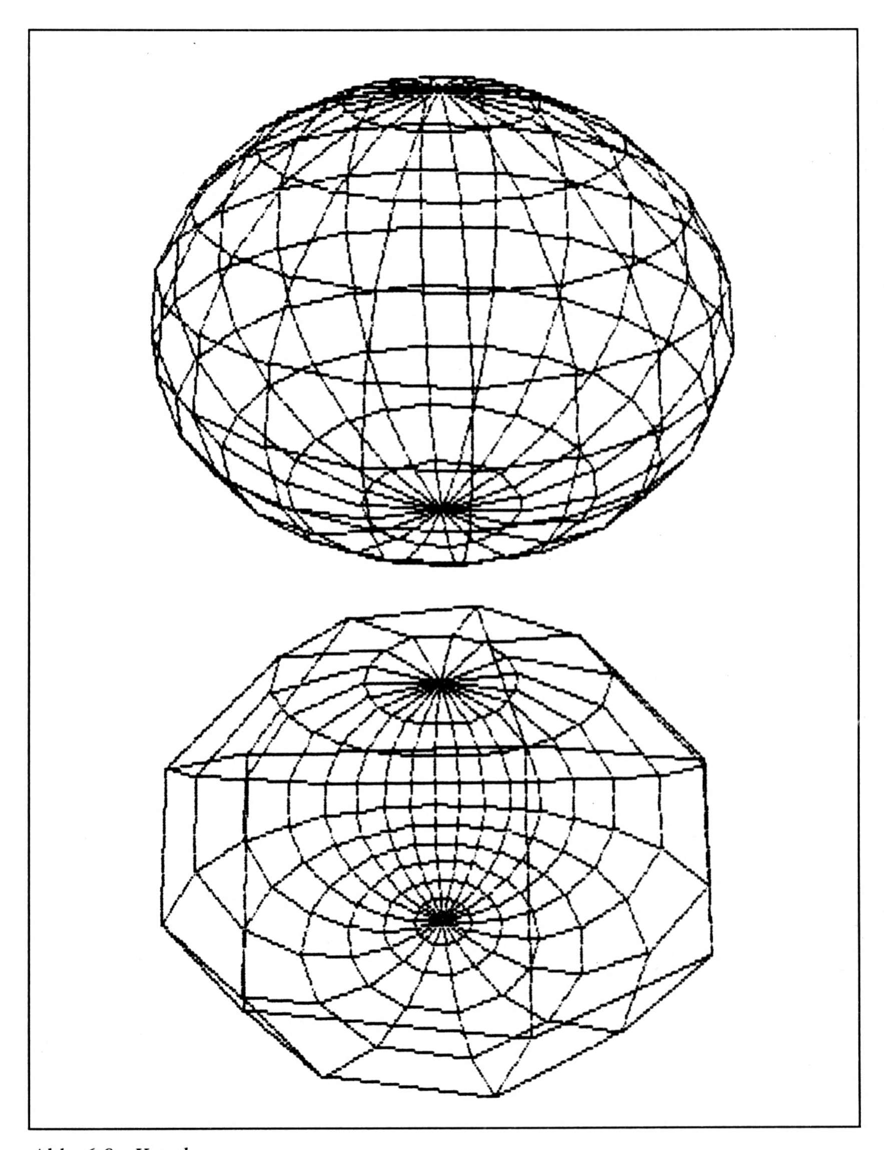


Abb. 6.9: Kugel

zung ist, daß die Punkte eine günstige Numerierung erhalten. Die Zählvariable muß dazu in der Schleife der untersten Verschachtelungsebene hochgezählt werden.

Die Zuordnung der Anfangs- und Endpunkte (ap(i), ep(i)) zu den Linien erfolgt durch eine weitere Schleife.

6.2 STEREOBILDER

Die bisher erzeugten Zentralprojektionen haben die Gegenstände immer so gezeigt, wie man sie sieht, wenn man ein Auge zukneift, da es sich immer nur um eine einzelne Projektion handelt. Das wirklich räumliche Sehen erfordert aber zwei Projektionen von verschiedenen Projektionszentren aus. Unsere Augen sehen von einem betrachteten Gegenstand zwei unterschiedliche Bilder.

Die beiden Bilder, die das rechte und das linke Auge erzeugen, unterscheiden sich um so mehr, je geringer der Betrachtungsabstand ist. Das können Sie leicht feststellen, indem Sie beim Betrachten verschiedener Gegenstände abwechselnd das rechte und dann das linke Auge zukneifen.

Unser Gehirn verarbeitet die unterschiedlichen Bilder und schätzt aufgrund der Abweichungen den Betrachtungsabstand. Das geschieht natürlich alles, ohne daß es uns bewußt wird, und wir meinen „räumlich" zu sehen. Um mit Abbildungen ein räumlich erscheinendes Bild zu erhalten, müssen wir beiden Augen Projektionen von verschiedenen Standpunkten zuführen.

Diese Projektionen zu erzeugen, ist kein Problem. Wir müssen dazu nur das Unterprogramm „Zentralprojektion" leicht verändern, so daß es die zwei Projektionen mit um den „Augenabstand" verschobenen Projektionszentren erzeugt. Abb. 6.10 soll die Entstehung des Stereobildes etwas veranschaulichen.

Die genaue Herleitung der Projektionsformeln erfolgt etwas später. Zunächst ist noch ein anderes Problem zu bewältigen: Wie kann man den Augen die Bilder getrennt zuführen? Jedes Auge darf nur das Bild sehen, das vom entsprechenden Projektionszentrum aus erzeugt wurde. Zu diesem Zweck wurden verschiedene Verfahren entwickelt. Einmal gibt es die sogenannten Stereoskope, mit denen man Stereofotografien betrachten kann. Dabei werden den Augen nebeneinanderliegende Bilder über ein Spiegelsystem zugeführt. Wer ein Stereoskop besitzt, kann sich Hardcopys von Zentralprojektionen damit betrachten. Es empfiehlt sich dabei, die Linien der Hardcopy mit einem etwas stärkeren Stift nachzuziehen.

Die andere Methode, die für uns etwas einfacher zu realisieren, dafür aber we-

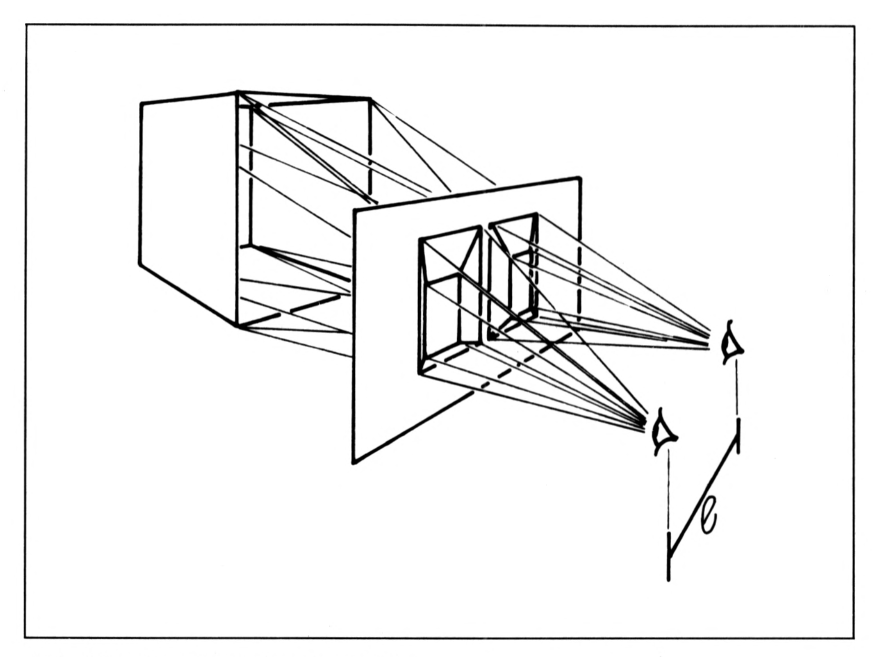


Abb. 6.10: Entstehung eines Stereobildes

niger wirkungsvoll ist, ist das Anaglyphenverfahren. Sicher kennen Sie die rot-grünen Brillen, die dazu verwendet werden, die Bilder zu trennen. Diese Brillen haben links ein rotes Glas (bzw. eine rote Folie), wodurch rote Gegenstände heller erscheinen und grüne dunkler, und rechts ein grünes Glas, das den gegenteiligen Effekt bewirkt.

Rote Linien auf hellem Grund sind für das linke Auge schlechter zu sehen, grüne dagegen gut. Bei dunklem Hintergrund ist es genau umgekehrt, was wir bei der Herstellung von Stereobildern berücksichtigen müssen.

Die rot-grünen Bilder lassen sich leicht auf dem Bildschirm erzeugen. Das vom weiter links liegenden Projektionszentrum erzeugte Bild wird rot gezeichnet, das andere grün. Dabei verwenden wir einen schwarzen Hintergrund. Wenn Sie keinen Farbmonitor haben, können Sie vom Bildschirminhalt eine Hardcopy herstellen und die Bilder mit transparentem Papier und Filzstift abzeichnen. Die Farben müssen Sie jetzt vertauschen (links Grün, rechts Rot), da das verwendete Papier hell ist und die Figur dunkel erscheint.

Sollten Sie keine solche Brille besitzen, dürfte es keine Schwierigkeiten bereiten, aus Pappe und bunter Kunststoffolie etwas Passendes zu basteln.

Doch nun zur Berechnung der Projektionen. Wie Sie in Abb. 6.10 sehen, müssen wir zuerst das Projektionszentrum etwas nach links verschieben und die Projektion in roter Farbe zeichnen (auf dunklem Grund), dann wird das Projektionszentrum nach rechts verschoben, und die Figur wird in Grün abgebildet. Abb. 6.11 soll uns bei der Herleitung der Formeln etwas unterstützen. Die Formel für die z-Koordinate bleibt gleich, da beide Projektionszentren immer noch auf der gleichen Höhe liegen. Bei der Berechnung der Bild-x-Koordinaten ergibt sich zunächst eine andere Steigung für den Projektionsstrahl. Bei der einfachen Zentralprojektion war die Steigung

m = x(i)/c

mit c = b+y(i) (b = Abstand des Beobachtungsstandortes vom Koordinatenursprung; siehe auch Abb. 6.5).

Den Abstand des Projektionszentrums von der Koordinatenachse wollen wir mit e/2 bezeichnen. e ist dann der Abstand der Projektionszentren untereinander. Es ergeben sich zwei Steigungsformeln für die zwei Projektionszentren.

links:

m = (x(i)+e/2)/c

rechts:

m = (x(i)−e/2)/c

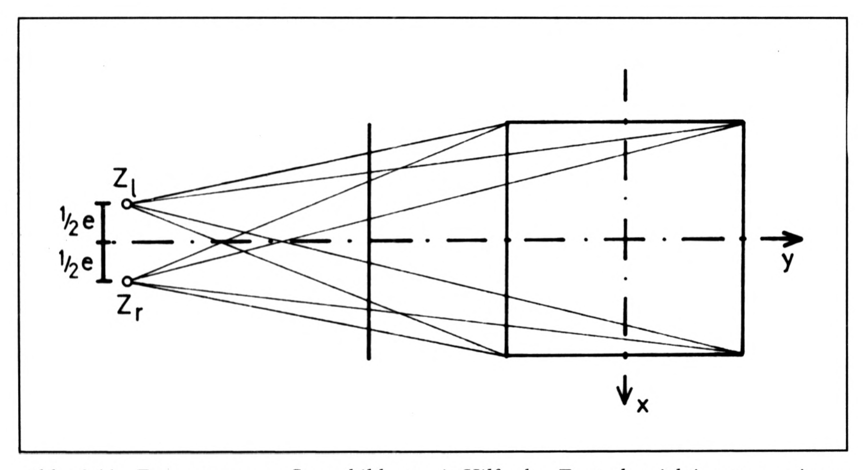


Abb. 6.11: Erzeugung von Stereobildern mit Hilfe der Zentralprojektion an zwei verschiedenen Projektionszentren

Außerdem müssen wir bei der Berechnung der Bildkoordinaten noch die Abweichung des Projektionszentrums hinzuzählen, da der Projektionsstrahl mit der Steigung m auf der Höhe des Beobachtungsstandpunktes schon eine Entfernung e/2 von der y-Achse hat. Damit erhalten wir folgende Formeln für die Bildkoordinaten:

links: $b(i) = (x(i)+e/2)/c*a-e/2$ (a ist die Brennweite)

rechts: $b(i) = (x(i)-e/2)/c*a+e/2$

Das sind alle Grundlagen, die wir brauchen, um das folgende Programm zu entwickeln. Es ist im wesentlichen aus einzelnen Teilen der 3D-Grafik-Unterprogrammsammlung zusammengestellt. Als Beispielfigur dient der Würfel von Abb. 6.1, der um einige Linien ergänzt wurde. Von dem Würfel wird ein Stereobild hergesellt, das jeweils nach ein paar Sekunden um 10 Grad weitergedreht wird. Wird eine Taste gedrückt, hält das Programm an und fragt nach Brennweite, Beobachtungsstandpunkt (negativ eingeben) und Augenabstand (womit der Abstand der Projektionszentren gemeint ist). Mit den Werten muß man etwas experimentieren, um einen einigermaßen räumlichen Eindruck zu erzeugen (zum Beispiel: Beobachtungsstandpunkt = −300; Brennweite = 250; Augenabstand = 10). Auch die Helligkeitseinstellung des Monitors muß eventuell reguliert werden. Eine möglichst dunkle Einstellung brachte bei unseren Versuchen die besten Ergebnisse. Außerdem war der Raum etwas verdunkelt.

```
1000 REM -----------------------------
1010 REM ---       Anaglyphen      ---
1020 REM ---     Demo-Programm     ---
1030 REM -----------------------------
1040 :
1050 DIM ap(100),ep(100)
1060 DIM x1(100),y1(100),z1(100)
1070 DIM x2(100),z2(100)
1080 :
1090 MODE 1
1100 PRINT CHR$(23);CHR$(3);
1110 INK 0,0:INK 1,6:INK 2,9:INK 3,13
1120 ORIGIN 320,200
1130 WINDOW #1,1,40,22,25
1140 PEN #1,3
1150 BORDER 0
1160 :
1170 DEG
1180 alpha=10
1190 :
1200 REM -----------------------------
1210 REM ---  Figurdaten einlesen  ---
1220 REM -----------------------------
```

Programm 6.2: 3D-Grafik

```
1230 :
1240 RESTORE
1250 :
1260 READ n :REM  Anzahl der Eckpunkte
1270 :
1280 FOR i=1 TO n
1290    READ x1(i),y1(i),z1(i):REM Koord.
1300 NEXT
1310 :
1320 READ k :REM     Anzahl der Linien
1330 :
1340 FOR i=1 TO k
1350    READ ap(i),ep(i):REM Anf. Ende
1360 NEXT
1370 :
2000 REM -----------------------------
2010 REM ---      Hauptprogramm      ---
2020 REM -----------------------------
2030 :
2040 CLS #1
2050 INPUT #1,"Beobachtungsstandort ";b
2060 INPUT #1,"Brennweite ";a
2070 INPUT #1,"Augenabstand ";e
2080 :
2090 CLS
2100 :
2110 GOSUB 3050
2120 GOSUB 4040
2130 :
2140 a$=INKEY$
2150 IF a$="" THEN 2090
2160 GOTO 2040
2170 :
3000 REM -----------------------------
3010 REM ---    Zentralprojektion    ---
3020 REM -----------------------------
3030    :
3040 REM ---    zwei Projektionen    ---
3050    farbe=0
3060    FOR s=-e/2 TO e/2 STEP e
3070       farbe=farbe+1
3080       :
3090 REM --- Projektion berechnen  ---
3100       :
3110       FOR i=1 TO n
3120          c=-b+y1(i)
3130          z2(i)=z1(i)*a/c
3140          x2(i)=(x1(i)-s)*a/c+s
3150       NEXT
3160       :
3170 REM ---       Darstellen        ---
3180       :
3190       FOR i=1 TO k
3200          MOVE x2(ap(i)),z2(ap(i))
3210          DRAW x2(ep(i)),z2(ep(i)),farbe
3220       NEXT
3230    :
3240    NEXT
```

Programm 6.2: 3D-Grafik (Forts.)

```
3250    :
3260 RETURN
3270 :
4000 REM ----------------------------
4010 REM ---       Drehung um Z     ---
4020 REM ----------------------------
4030    :
4040    FOR i=1 TO n
4050       x=x1(i)
4060       y=y1(i)
4070       x1(i)=x*COS(alpha)-y*SIN(alpha)
4080       y1(i)=x*SIN(alpha)+y*COS(alpha)
4090    NEXT
4100    :
4110 RETURN
4120 :
5000 REM ----------------------------
5010 REM ---       Figurdaten       ---
5020 REM ----------------------------
5030 :
5040 DATA 8
5050 DATA 100,100,100,-100,100,100,-100,-100,100,100,-10
0,100
5060 DATA 100,100,-100,-100,100,-100,-100,-100,-100,100,
-100,-100
5070 :
5080 DATA 18
5090 DATA 1,2,2,3,3,4,4,1,5,6,6,7,7,8,8,5,1,5,2,6,3,7,4,
8
5100 DATA 1,3,2,4,1,7,4,6,5,7,6,8
```

Programm 6.2: 3D-Grafik (Forts.)

Variablenübersicht:

ap(i), ep(i):	Anfangs- und Endpunkte der Linien
x1(i), y1(i), z1(i):	Raumkoordinaten
x2(i), z2(i):	Bildkoordinaten
alpha:	Drehwinkel
n:	Anzahl der Eckpunkte
k:	Anzahl der Linien
b:	Beobachtungsstandort auf der y-Achse
a:	Brennweite
e:	Augenabstand (= Abstand der Projektionszentren)
a$:	Hilfsvariable für Tastaturabfrage
s:	Schleifenvariable für Erzeugung verschiedener Projektionen
farbe:	gibt Zeichenstift an; wird in der Schleife hochgezählt und ergibt eine rote Zeichnung für das linke und eine grüne für das rechte Auge
weitere Variablen:	siehe vorangegangene Unterprogrammsammlung

Programmbeschreibung

Zeilen 1050 bis 1070:	Dimensionieren der Felder für die Anfangs- und Endpunkte der Linien, die Raum- und die Bildkoordinaten.
Zeile 1090:	Mode 1 = Vierfarbenmodus.
Zeilen 1100, 1110:	Anweisung zur Verarbeitung des DRAW-Befehls. Die Pixels, die sich bereits auf dem Bildschirm befinden, werden mit neuen Pixels durch das logische OR verknüpft. Die Farbe Rot hat die Nummer 1 (&01), Grün hat die Nummer 2 (&10). Schneiden sich nun Linien der zwei Projektionen, so müssen die Schnittpunkte für beide Augen sichtbar sein. Die Verknüpfung von &01 (Rot) und &10 (Grün) durch OR ergibt &11 (3). Die Farbe mit Nummer 3 ist Weiß, also für beide Augen sichtbar.
Zeile 1120:	Koordinatenursprung festlegen.
Zeile 1130:	Eingabefenster festlegen.
Zeile 1140:	Schriftfarbe.
Zeile 1150:	Randfarbe.
Zeile 1170:	Auf Gradmaß umschalten.
Zeile 1180:	Drehwinkel festlegen.
Zeilen 1200 bis 1300:	Siehe entsprechenden Teil der Unterprogrammsammlung.
Zeilen 2000 bis 2170:	Hauptprogramm; Eingabe der Projektionsdaten. In Zeile 2110 wird zum Projektionsteil gesprungen, in Zeile 2120 zu dem Teil, der die Drehung ausführt. In Zeile 2140 wird geprüft, ob eine Taste gedrückt wurde. Wenn nicht, dann läuft das Programm normal weiter; sonst wird wieder zum Eingabeteil verzweigt.
Zeilen 3000 bis 3270:	Die Zeilen 3090 bis 3220 sind aus der Unterprogrammsammlung entnommen und berechnen und zeichnen die Projektion. Nur der DRAW-Befehl in Zeile 3210 ist um die Farbangabe ergänzt, und in Zeile 3140 steht die schon beschriebene geänderte Formel für die x-Koordinaten. Das ganze Programm ist

in eine Schleife eingebunden, die in zwei Durchläufen Farbe und Projektionszentrum vorgibt.

Zeilen 4000 bis 4110: Drehung um z-Achse, siehe entsprechenden Teil der Unterprogrammsammlung.

Zeilen 5000 bis 5100: Figurdaten.

Zum Schluß einige Stereobilder (Abb. 6.12).

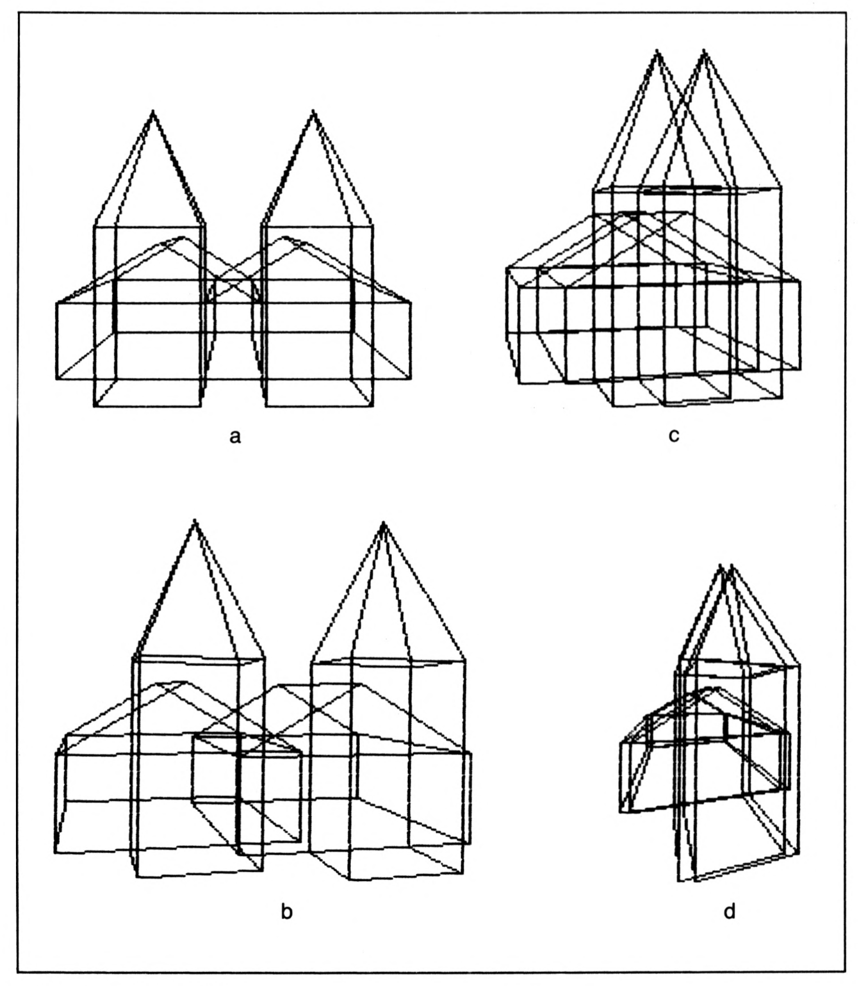


Abb. 6.12: Stereobilder

6.3 DARSTELLUNG RÄUMLICHER FUNKTIONEN

Mathematische Funktionen, die von zwei Parametern (Variablen; x, y) abhängen, kann man als räumliche Flächen interpretieren. Hier bietet sich die grafische Auswertung solcher Funktionen mit dem Rechner an. Bei geeigneter Wahl von Abbildungsmaßstab und Betrachtungswinkeln ergeben sich mit der senkrechten Parallelprojektion (Grund-, Aufriß) recht anschauliche Funktionsbilder. Doch während man sich bei der Darstellung von Polyedern gerade noch vorstellen konnte, welche Linien im Vorder- und welche im Hintergrund liegen, ist es bei den Funktionsbildern meistens unmöglich, den Überblick zu behalten.

Es ist also notwendig, verdeckte Punkte zu erkennen und aus der Darstellung zu entfernen. So schwierig das bei den Polyedern ist, so einfach ist es bei den (stetigen) Funktionen. Voraussetzung dabei ist, daß die darzustellende Fläche geschlossen ist, also keine Löcher hat.

In Abb. 6.13 ist eine solche Funktion dargestellt. Aus der gesamten Funktionslandschaft (die unendlich sein kann) wird ein Block herausgeschnitten, der mit Hilfe des Rechners abgebildet werden soll. An den Schnittkanten, die

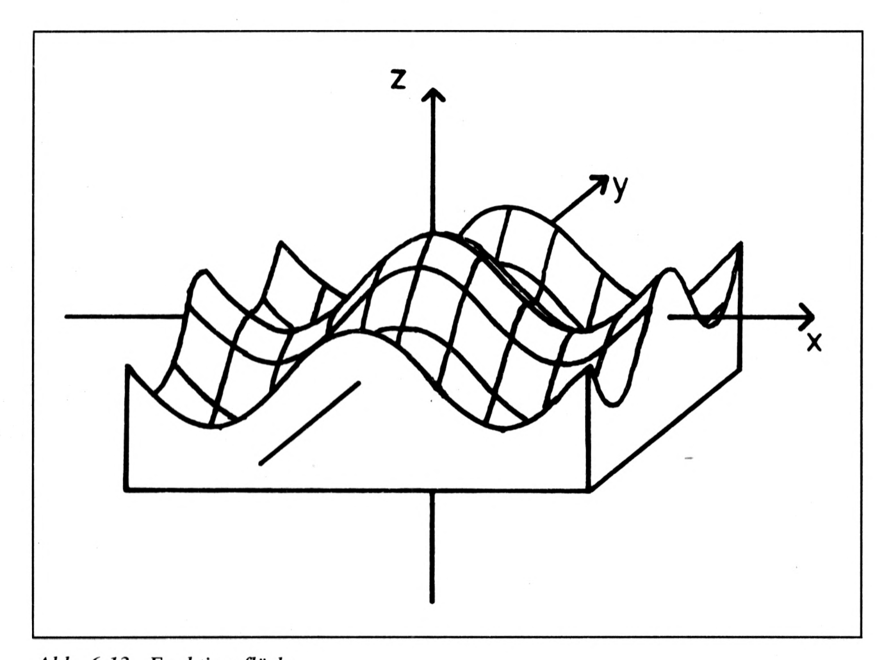


Abb. 6.13: Funktionsfläche

parallel zur x- bzw. y-Achse verlaufen, erhalten wir vier Funktionskurven, die nur von einer Variablen abhängen (x oder y). Zwei dieser Funktionskurven zeigen immer zum Betrachter; sie sind also in jedem Fall sichtbar.

Beim Berechnen und Abbilden der Funktionsfläche gehen wir immer zeilenweise vor, d. h. zunächst wird für y der kleinste Wert, der noch im Abbildungsbereich liegt, gewählt, und x läuft vom kleinsten zum größten Wert des Abbildungsbereichs. Wir beginnen immer im Vordergrund. Damit erhalten wir die erste Funktionskurve, die an der Vorderseite des herausgeschnittenen Blocks liegt. Dann wird y um eine Schrittweite erhöht und die nächste Kurve berechnet.

Die Punkte der zweiten Kurve können in der Abbildung über, unter oder auf der alten Kurve liegen. In den ersten beiden Fällen sind die Punkte sichtbar und können eingezeichnet werden. Der Bereich zwischen zwei auf dem Bildschirm übereinanderliegenden Punkten ist undurchsichtig, da die Fläche keine Löcher haben soll, und muß für weitere Punkte gesperrt werden. Um die zu sperrenden Bereiche zu erfassen, teilen wir die Zeichenfläche, also den Bildschirm, in Spalten auf.

Beim CPC haben wir 640 Spalten mit je 200 Punkten. Für jede Spalte reservieren wir zwei Variablen, die als Zeiger auf den höchsten und den tiefsten in dieser Spalte gezeichneten Punkt dienen. Bevor die Funktion durch das Hauptprogramm (in zwei geschachtelten Schleifen) gezeichnet werden kann, müssen die vorne liegenden Kurven (Abb. 6.14) erfaßt werden. Dazu verwenden wir zwei FOR...NEXT-Schleifen.

Zuerst wird y auf den kleinsten Wert gesetzt. x läuft vom kleinsten bis zum größten Wert. In der Schleife wird der jeweilige Funktionswert z berechnet, der Punkt (x,z) (y entfällt wegen der Projektion auf den flachen Bildschirm) gezeichnet, und die Zeiger u(x), o(x) werden gleich z gesetzt. Dann wiederholt sich das Ganze mit vertauschten Variablen. x wird auf den kleinsten Wert gesetzt, und y wird als Laufvariable verwendet. In Abb. 6.14 ist eine Bildschirmspalte mit den Zeigern o(i) und u(i) eingezeichnet. Auf den folgenden Abbildungen 6.15 und 6.16 sehen Sie, wie sich das Funktionsbild weiterentwickelt. Jetzt wird mit zwei verschachtelten Schleifen für die Koordinaten x und y weitergezeichnet. Immer wenn ein neuer Punkt berechnet wird, muß überprüft werden, ob er sichtbar ist, bevor er gezeichnet wird.

```
2340 W=O
2350 IF z>o(x) THEN o(x)=z:W=1
2360 IF z<u(x) THEN u(x)=z:W=1
2370 IF W=1 THEN PLOT x,z
```

Außerdem muß, falls er sichtbar ist, der entsprechende Zeiger versetzt werden, was im oben gezeigten Programmausschnitt schon berücksichtigt wurde.

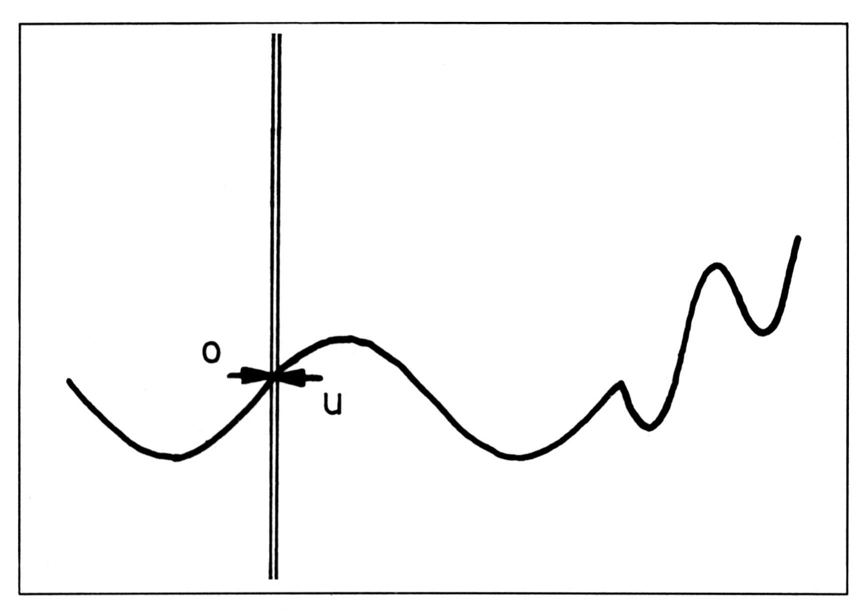


Abb. 6.14: Zeichnen einer Funktionsfläche (Teil 1)

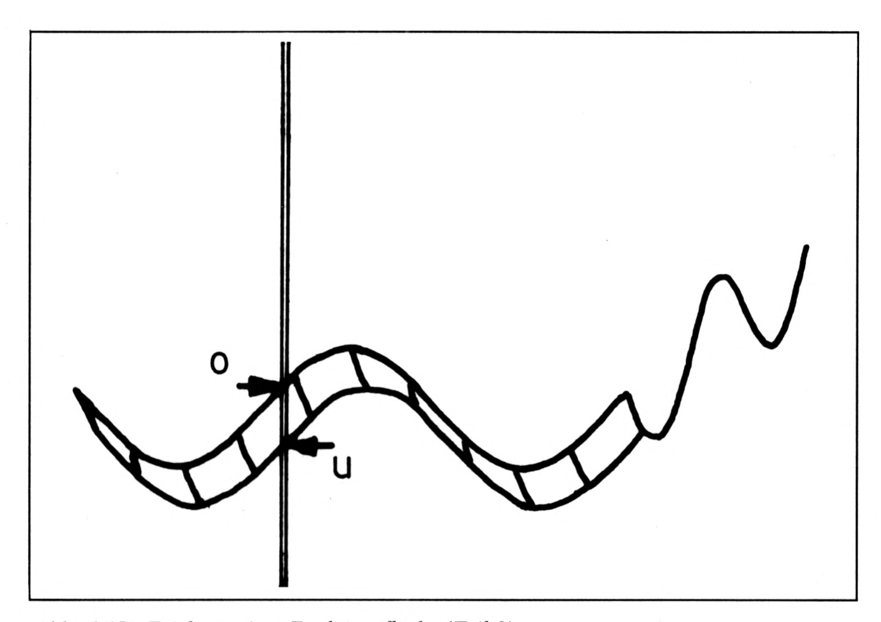


Abb. 6.15: Zeichnen einer Funktionsfläche (Teil 2)

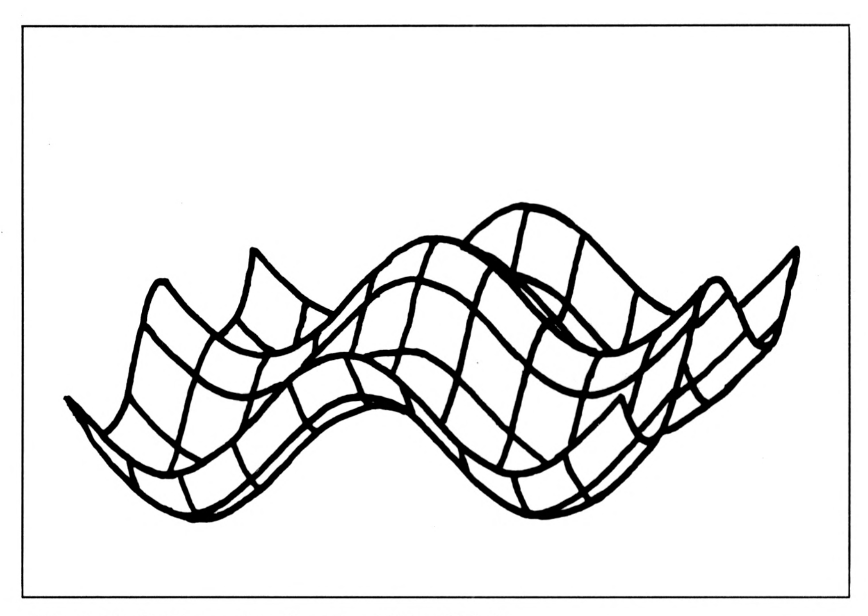


Abb. 6.16: Zeichnen einer Funktionsfläche (Teil 3)

Das Programm hat noch einen kleinen Schönheitsfehler. Es liefert einen Punkt dicht neben dem anderen, so daß hinterher die ganze Fläche ausgefüllt ist und von der Funktion nur noch die Umrisse zu erkennen sind. Durch einen kleinen Trick läßt sich erreichen, daß wie in Abb. 6.16 das x/y-Koordinatensystem auf die Funktionsfläche projiziert wird. Dabei wird einfach nur jeder zehnte Punkt wirklich gezeichnet. Die letzte Zeile des eben gezeigten Programmausschnitts sieht dann so aus:

```
2370 IF W=1 THEN
         IF (x/10=INT(x/10)) OR (y/10=INT(y/10))
         THEN PLOT x,z
```

Natürlich wollen wir die Funktion nicht genau von der Seite betrachten, sondern einen günstigen Blickwinkel wählen bzw. die Funktionsfläche in eine günstige Lage drehen. Dazu verwenden wir die bereits beschriebenen Formeln. Zunächst drehen wir die Funktionsfläche um die x-Achse, so daß wir sie schräg von oben oder unten sehen können, um den Winkel gamma:

```
x1 = x
y1 = y*cos(gamma)-z*sin(gamma)
z1 = y*sin(gamma)+z*cos(gamma)
```

und dann um die z-Achse um den Winkel alpha:

x2 = x1*cos(alpha)−y1*sin(alpha)
y2 = x1*sin(alpha)+y1*cos(alpha)
z2 = z1

Aus Rechenzeitgründen fassen wir die Formeln noch etwas zusamen:

x2 = x*cos(alpha)−(y*cos(gamma) −z*sin(gamma))*sin(alpha)
y2 wird nicht benötigt
z2 = y*sin(gamma)+z*cos(gamma)

Diese zwei Formeln liefern die Bildschirmkoordinaten x2, z2 aus den Raumkoordinaten x, y, z.

Im Programm werden die Winkel α und γ vorgewählt, und die Drehung muß dann für jeden Punkt extra berechnet werden, weshalb dieser Teil als Unterprogramm ausgeführt wird. Wenn, wie im folgenden Beispiel, ein Funktionsausschnitt von 100 mal 100 Punkten berechnet wird, wird dieses Unterprogramm 10000mal aufgerufen. Es enthält jedoch noch viele Winkelfunktionen (sin, cos), die viel Rechenzeit benötigen. Da die Winkel während des Programmablaufs nicht mehr geändert werden, können wir die Werte der Winkelfunktionen am Programmanfang berechnen, diese in Variablen ablegen und die Winkelfunktionen selbst aus den Formeln eliminieren, um einiges an Rechenzeit zu sparen:

```
1280 c1=sin(alpha)
1290 c2=cos(alpha)
1300 c3=sin(gamma)
1310 c4=cos(gamma)
```

Die Drehformeln:

```
3060 x2=x*c2-(y*c4-z*c3)*c1
3070 z2=y*c3+z*c4
```

Jetzt können wir anfangen, ein Programm zu entwerfen. Die Funktion, die gezeichnet werden soll, definieren wir in einem Unterprogramm, damit sie leicht ausgetauscht werden kann. Die Funktion kann dann sogar aus einem ganzen Programm mit Verzweigungen bestehen. Das folgende Beispielprogramm (Programm 6.3) zeichnet einen quadratischen Ausschnitt der Funktion, in dessen Mitte der Koordinatenursprung liegt. Die x- und y-Koordinaten laufen von −100 bis 100. Da bei diesem Ausschnitt 40000 Schleifendurchläufe nötig sind, um den Funktionsgraphen zu berechnen, müssen Sie sich etwa eine Stunde gedulden, bis das Programm beendet ist.

```
1000 REM ------------------------------
1010 REM ---      3-D Funktionen      ---
1020 REM ------------------------------
1030 :
1040 REM ---      Voreinstellung      ---
1050 :
1060 MODE 2
1070 INK 0,24:INK 1,4
1080 BORDER 4
1090 ORIGIN 320,200
1100 DEG
1110 :
1120 REM ---       Zeigerfelder       ---
1130 :
1140 DIM o(640),u(640)
1150 :
1160 REM ------------------------------
1170 REM ---        Parameter         ---
1180 REM ------------------------------
1190 :
1200 REM ---  Winkel f. Dreh. um Z   ---
1210 :
1220 alpha=30
1230 :
1240 REM ---  Winkel f. Dreh. um Y   ---
1250 :
1260 gamma=15
1270 :
1280 c1=SIN(alpha)
1290 c2=COS(alpha)
1300 c3=SIN(gamma)
1310 c4=COS(gamma)
1320 :
2000 REM ------------------------------
2010 REM ---       Vordergrund-       ---
2020 REM ---    kurven bestimmen      ---
2030 REM ------------------------------
2040 :
2050 x=-100
2060 FOR y=-100 TO 100
2070    GOSUB 4040
2080    GOSUB 3060
2090    n=320+INT(x2)
2100    o(n)=z2
2110    u(n)=z2
2120    PLOT x2,z2
2130 NEXT
2140 :
2150 y=-100
2160 FOR x=-99 TO 100
2170    GOSUB 4040
2180    GOSUB 3060
2190    n=320+INT(x2)
2200    o(n)=z2
2210    u(n)=z2
2220    PLOT x2,z2
2230 NEXT
```

Programm 6.3: Dreidimensionale Darstellung von Funktionsflächen

```
2240 :
2250 REM ------------------------------
2260 REM ---      Hauptschleife     ---
2270 REM ------------------------------
2280 :
2290 FOR y=-99 TO 100
2300    FOR x=-99 TO 100
2310       GOSUB 4040
2320       GOSUB 3060
2330       n=320+INT(x2)
2340       w=0
2350       IF z2>o(n) THEN o(n)=z2:w=1
2360       IF z2<u(n) THEN u(n)=z2:w=1
2370       IF w=1 THEN IF (x/10=INT(x/10)) OR (y/10=INT(
y/10)) THEN PLOT x2,z2
2380    NEXT
2390 NEXT
2400 :
2410 END
2420 :
3000 REM ------------------------------
3010 REM ---      Unterprogramme    ---
3020 REM ------------------------------
3030 :
3040 REM ---   Koordinatendrehung   ---
3050    :
3060    x2=x*c2-y*c1
3070    z2=(x*c1+y*c2)*c3+z*c4
3080    :
3090 RETURN
3100 :
4000 REM ------------------------------
4010 REM --- Hier Funktion eingeben ---
4020 REM ---     zum Beispiel:      ---
4030 REM ------------------------------
4040    :
4050    z=50*(COS(2*x)+COS(2*y))
4100 RETURN
```

Programm 6.3: Dreidimensionale Darstellung von Funktionsflächen (Forts.)

Variablenübersicht:

o(i):	Zeiger für „oben“
u(i):	Zeiger für „unten“
alpha:	Drehwinkel für Drehung um z-Achse
gamma:	Drehwinkel für Drehung um x-Achse
c1...c4:	Konstanten für Drehung
x, y, z:	Raumkoordinaten
x2, z2:	Bildschirmkoordinaten
n:	Spaltenindex für Zeigervariablen
w:	Flag für Sichtbarkeit

Programmbeschreibung

Zeile 1060:	Grafikmodus 2 (640*2000 Punkte) wählen.
Zeilen 1070, 1080:	Farben wählen.
Zeile 1090:	Koordinatenursprung in die Bildschirmmitte verlegen.
Zeile 1100:	Von Bogen- in Gradmaß umschalten (Vollkreis = 360 Grad). Nachdem die Koordinaten für die Drehung ermittelt sind, kann, wenn es die zu zeichnende Funktion erfordert, wieder auf Bogenmaß zurückgeschaltet werden (RAD in Zeile 1310 einfügen).
Zeile 1140:	Zeigerfelder dimensionieren.
Zeile 1220:	Drehwinkel für Drehung um z-Achse festlegen. Kann durch INPUT ersetzt werden.
Zeile 1260:	Wie oben; Drehung um x-Achse.
Zeilen 1280 bis 1310:	Konstanten für Drehung ermitteln.
Zeilen 2050 bis 2130:	Bestimmen der vollständig sichtbaren Kurven in der Ebene x = −100.
Zeilen 2150 bis 2230:	Das gleiche für die Ebene y = −100.
Zeilen 2290 bis 2390:	Die Hauptschleife ist schon im Text beschrieben.
Zeilen 3000 bis 3090:	Unterprogramm für Koordinatendrehung.
Zeilen 4040 ff.:	Hier können Sie die Funktion festlegen.

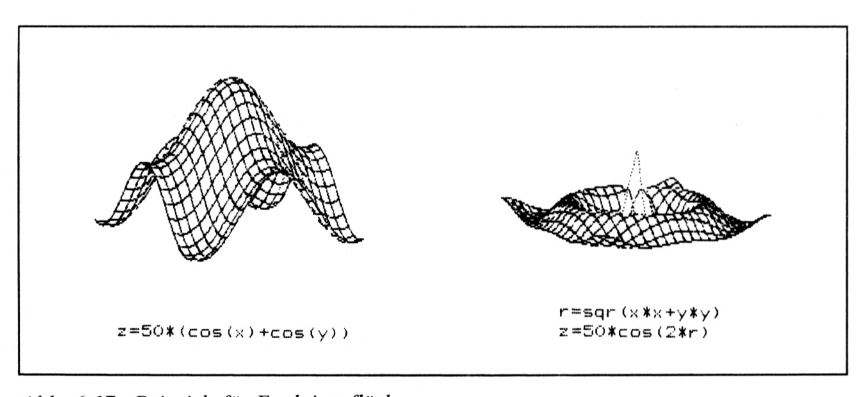


Abb. 6.17: Beispiele für Funktionsflächen

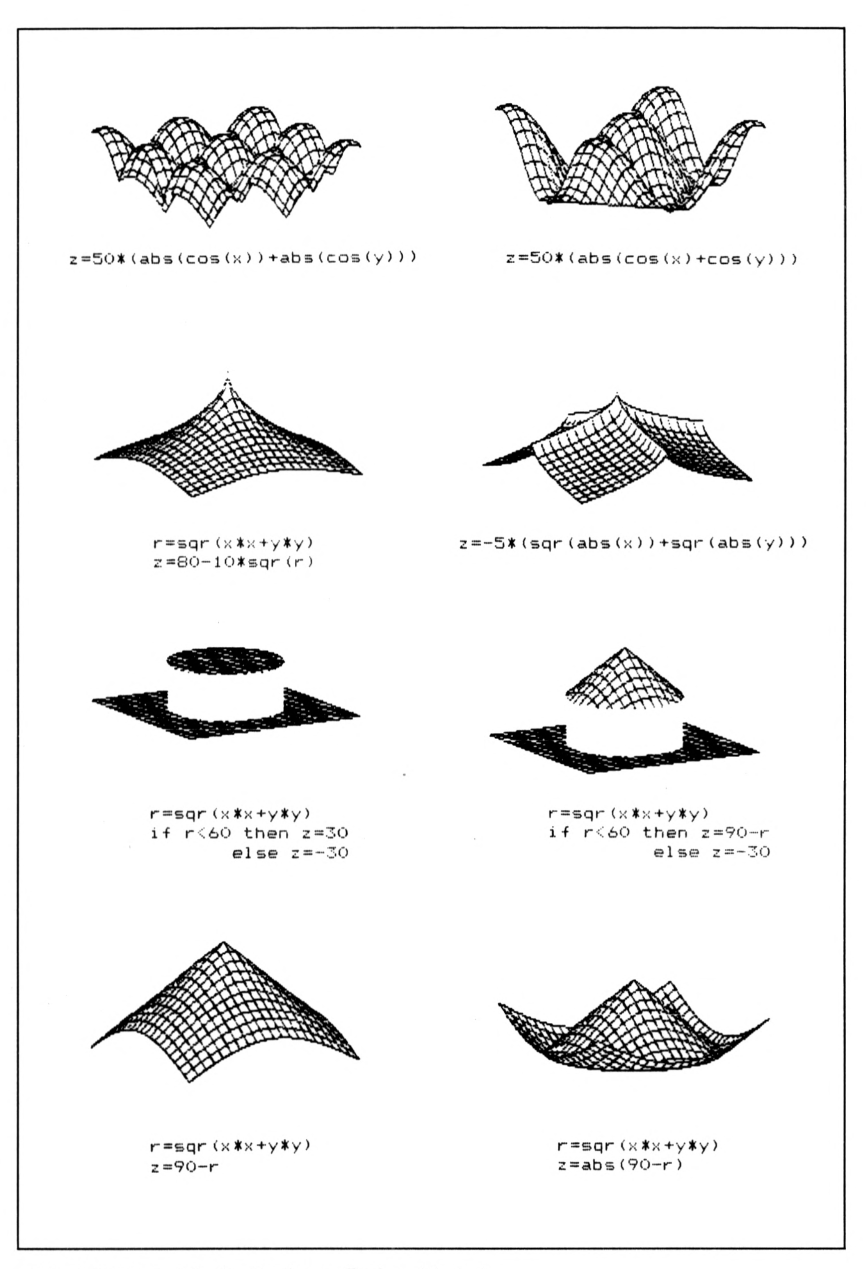


Abb. 6.17: Beispiele für Funktionsflächen (Forts.)

Kapitel 7
Grafiken und Speichermedien

Trotz der schnellen Grafikbefehle und auch trotz unserer in Kapitel 2 vorgestellten Befehlserweiterung dauert es bei manchen Grafiken doch sehr lange, bis sie erstellt sind. Hier wären z. B. die in Kapitel 6 vorgestellten dreidimensionalen Funktionen zu nennen. Wenn man diese Grafiken häufiger braucht oder sie eventuell zu Vorführzwecken dienen sollen, ist es eine sehr mühselige Angelegenheit, diese jeweils neu zu erstellen. Aber auch hier ist im Befehlsvorrat vorgesorgt.

Zunächst das Speichern: Es wird wie bei Maschinenprogrammen der SAVE-Befehl herangezogen, jedoch sind hier außer dem Dateinamen – für die Grafikdaten – weitere Parameter zu berücksichtigen. Gibt man, durch Komma getrennt, hinter dem Namen ein B an, so werden die Daten, wie sie im RAM stehen, auf die Diskette/Kassette übertragen. Da man jedoch nicht den gesamten RAM-Bereich abspeichern möchte, ist es zweckmäßig, noch den betreffenden Bereich anzugeben. Dies geschieht mit zwei weiteren Parametern beim SAVE-Befehl, die zweckmäßigerweise in hexadezimaler Schreibweise angegeben werden. Sie werden, ebenfalls durch Kommata getrennt, an die vorherigen Parameter angehängt, wobei der erste Parameter den Beginn des Speicherbereichs und der zweite Parameter die Länge desselben angibt.

Da der Bereich, in dem die Grafik im Speicher abgelegt ist, identisch mit dem Bildschirm-RAM ist, ist es zweckmäßig, den SAVE-Befehl aus einem Programm heraus aufzurufen. Dadurch wird der sonst einzugebende SAVE-Befehl auf dem Bildschirm unterdrückt und somit auch nicht mitgespeichert.

Bleiben nur noch die Meldungen des Computers, die im Normalfall dem Anwender den korrekten Ablauf des Speichervorgangs anzeigen. Diese Meldungen kann man unterdrücken, wenn man dem Dateinamen ein Ausrufezeichen voranstellt.

Der Bildschirmspeicher befindet sich im RAM ab der Adresse &C000 und hat eine Länge von 16 KByte, was in hexadezimaler Schreibweise &4000 ent-

spricht. Eine Grafik kann also gespeichert werden, wenn Sie eingeben:

```
SAVE !'Name',B,&C000,&4000
```

Der Ladevorgang verläuft analog dem Laden eines Programms, er sollte jedoch zweckmäßigerweise auch aus einem Programm bei vorher gelöschtem Bildschirm gestartet werden. Bei Aufruf des Namens – aus eben genannten Gründen wieder mit Ausrufezeichen – erkennt der Rechner selbst, daß es sich um eine Binärdatei handelt, und lädt die Daten genau an die Stelle, aus der sie vorher gespeichert wurden. Diesen Vorgang kann man übrigens nicht nur mit Grafiken durchführen, sondern auch bei Maschinenprogrammen, wobei sich der Rechner selbst einen Vermerk in die Datei schreibt, welcher Speicherbereich gesichert wurde.

Haben Sie Ihre Grafik von einer anderen Stelle aus auf Diskette/Kassette gesichert, was mit Hilfe eines Maschinenprogramms möglich ist, so ist nach dem Ladevorgang noch die Anfangsadresse im RAM anzugeben, ab der die Daten abgelegt werden sollen. Fassen wir noch einmal zusammen: Mit

```
LOAD !'Name'
```

kann eine Grafikdatei, die mit SAVE !'Name',B,&C000,&4000 gespeichert wurden, sehr einfach wieder geladen werden. Mit

```
LOAD!'Name',&C000
```

kann eine Grafikdatei in das Bildschirm-RAM geladen werden, die von einem anderen Speicherbereich aus auf das externe Speichermedium gesichert wurde.

Manchmal ist auch ein Mischen von zwei gespeicherten Grafiken sinnvoll und wünschenswert. Mit den vorgenannten Befehlen läßt sich dies nicht realisieren. In diesem Fall müssen die Daten zunächst mit PEEK aus dem RAM ausgelesen werden, um anschließend als sequentielle Datei auf einer Diskette/Kassette Platz zu finden. Durch ein entsprechendes Programm mit den gewünschten Verknüpfungen (UND, ODER, XOR) und gleichzeitiges Lesen von zwei und mehr sequentiellen Dateien können dann die Grafiken auf dem Bildschirm gemischt werden.

Kapitel 8
Sprites

Sprites sind in ihrer Definition nicht vollkommen festgelegt. Dem allgemeinen Gebrauch in der Computertechnik nach handelt es sich dabei um frei definierbare oder auch vorgegebene Zeichen. Diese können eine feste Größe haben, müssen aber nicht. Ebenso kann die Darstellung je nach Computerart und Anwendungszweck in Farbe oder, meist bei höherer Auflösung des Bildschirms, auch einfarbig erfolgen. Diese Zeichen sollen frei am Bildschirm zu positionieren sein und nach Möglichkeit den Hintergrund nicht zerstören.

Sprites kommen eigentlich aus dem Telespielbereich und werden heute ebenfalls fast nur bei Computerspielen eingesetzt. Bisher nur sehr sparsam, aber immer häufiger werden sie auch bei menüunterstützten Anwenderprogrammen eingesetzt. Dort sollen sie, bewegt mit Hilfe einer Maus oder einer Art von Joystick, helfen, den Benutzer durch das Programm zu führen.

8.1 ERZEUGUNG VON SPRITES

Die Erzeugung von Sprites kann auf verschiedenen Wegen erfolgen. Jeder dieser Wege stellt unterschiedliche Anforderungen an die Soft- und Hardware eines Computers.

Den softwaremäßig einfachsten Weg stellt die Verwendung eines Spezialschaltkreises (IC) dar. Solche ICs sind meist nicht allzu teuer, aber fast nur zum Einsatz als Videospiel-Controller gedacht.

Zum Spielen ist die erzeugte Grafik sehr gut geeignet, aber die Darstellung hochauflösender Grafik ist fast nie möglich. Dieses IC bewegt dann einen dafür vorgesehenen Speicherbereich, der den Bildausschnitt (das Sprite) enthält, in der gewünschten Richtung und der dafür bestimmten Geschwindigkeit über den Bildschirm.

Ein weiterer Weg besteht in der Verwendung eines relativ teuren Bildschirmprozessors. Diese Prozessoren sind vor allem für hochauflösende Grafiken gedacht. Sie enthalten zumeist einen fest eingebauten Zeichensatz und bieten die Möglichkeit, einen Teil des Bildspeichers zur Ablage von frei definierbaren Zeichen zur Verfügung zu stellen. Die Nachteile dieser Prozessoren sind,

abgesehen vom hohen Preis, daß sie vor allem für hohe und sehr hohe Bildauflösungen konzipiert wurden. Das mag zwar als ein Vorteil erscheinen, ist aber, wenn man die zur Zeit noch hohen Speicherplatzkosten bedenkt, ein Nachteil, denn der Bedarf an eigenem Bildspeicher ist enorm, selbst bei einer monochromen Auflösung von 1024*1024 Pixel (Bildpunkten) werden bereits 1 MBit, also 128 KByte, als reiner Bildspeicher benötigt. Ganz davon abgesehen sind Monitore mit solcher Auflösung in Farbe meist nicht unter 20 000 DM zu haben.

Den dritten und letzten Weg – der hier beschrieben werden soll – kann man beim CPC gehen. Der CPC ist mit einem sogenannten Video- oder auch Bildschirm-Controller ausgestattet. Dieser Baustein ist eigentlich zur reinen Zeichendarstellung ausgelegt, kann aber auch, mit Hilfe einer bestimmten Beschaltung, bis zu 512 KByte Speicher adressieren. Beim CPC adressiert er 16 KByte Speicher. Die Sprites können nicht vom Controller gesteuert werden, sondern müssen vom Mikroprozessor selbst auf den Bildschirm gebracht und bewegt werden.

Eine weitere Einschränkung ist bereits durch die Organisation der Bilderzeugung festgelegt. Die Sprites müssen nämlich in den Bildspeicher eingeschrieben werden. Dadurch entsteht der Nachteil, daß entweder das eingeschriebene Sprite den Hintergrund zerstört oder, bei geeigneter Einschreibweise, seine eigenen Farben und die des Hintergrundes verändert, sofern der Hintergrund eine andere Farbe als INK 0 enthält. Bei dieser Weise, das Sprite einzuschreiben, handelt es sich um die EXKLUSIV-ODER-Verknüpfung mit dem Hintergrund (vgl. Kapitel 5). Diese Verknüpfung stellt sicher, daß nach jedem zweiten Schreiben des Sprites auf der gleichen Stelle der Hintergrund wieder dem Original entspricht, denn durch die XOR-Verknüpfung mit sich selbst wird das Sprite wieder gelöscht.

Anwendung von Sprite-Routinen

Wie im vorangegangenen Abschnitt zu erfahren war, werden die Sprites mit Hilfe der XOR-Verknüpfung auf den Bildschirm gebracht. Bevor die Sprite-Routinen in Aktion treten können, müssen sie initialisiert werden. Wie die Einbindung der Befehle als Befehlserweiterung in das Betriebssystem bzw. hier in das BASIC geschieht, kann im Kapitel 2 nachgelesen werden. Nach der Initialisierung, die durch CALL &9000 vollzogen wird, stehen zwei neue Befehle zur Verfügung.

|*SPRITE,x,y,nummer*

Der erste Befehl benötigt, wie bereits ersichtlich, drei Parameter. Diese müssen, durch Kommata getrennt, an den Befehlsnamen angehängt werden. Zwi-

schen dem Namen und dem ersten Komma dürfen keine Leerzeichen stehen, da sonst der Befehl nicht erkannt wird. Die Parameter können Konstanten oder Variablen sein. Der erste Parameter gibt die X-Koordinate des linken unteren Punktes des Sprites an. Die Y-Koordinate desselben Punktes wird durch den zweiten Parameter dargestellt. Der letzte Parameter gibt die Sprite-Nummer an, die zwischen 0 und 15 liegen darf. Ist sie größer, so wird sie durch eine UND-Verknüpfung mit &0F auf einen gültigen Wert gebracht.

Die Koordinaten des Sprites beziehen sich immer auf die untere linke Ecke des beschreibbaren Bildschirms und werden wie normale Grafikkoordinaten angegeben. Sie müssen außerdem so gewählt werden, daß das Sprite innerhalb des Bildschirms dargestellt werden kann. Ansonsten wird das Sprite nicht auf den Bildschirm geschrieben, d. h. halbe Sprites am Rand des Bildschirms sind ebenfalls nicht möglich.

Liegen die Koordinaten außerhalb des zulässigen Bereichs, so kehrt die Routine zum aufrufenden BASIC zurück, ohne daß sie etwas am Bildschirm bewirkt hätte. Die Koordinaten werden von der SPRITE-Routine auf Gültigkeit überprüft. Der Gültigkeitsbereich für die X-Koordinaten liegt zwischen 0 und 576, der für die Y-Koordinaten zwischen 0 und 368. In BASIC-Programmen ist es nötig, sich die Koordinaten des Sprites in Variablen zu merken, um das Sprite wieder löschen zu können. Das Sprite wird dann nämlich durch einmaliges |SPRITE,x,y,nr mit dem Bildschirm EXKLUSIV-ODER-verknüpft.

Nur durch ein zweites Aufrufen dieser Routine mit den gleichen Parametern wird dieses Sprite durch die XOR-Verknüpfung wieder gelöscht. Achtung: Die Routine arbeitet nur im MODE 0 korrekt, in den anderen Modi wird das Sprite nicht richtig dargestellt.

Um also ein Sprite auf dem Bildschirm zu bewegen, muß es zuerst mit dem Befehl SPRITE dargestellt werden. Dann kann das Sprite durch Darstellen mit neuen veränderten Koordinaten ein Stück versetzt geschrieben werden, und das alte mit den alten Koordinaten kann gelöscht werden.

Mit dem Befehl können bis zu sechzehn verschiedene Sprites dargestellt werden, die Anzahl der gleichen Sprites auf dem Bildschirm ist theoretisch nur durch die Auflösung des CPC begrenzt. Ab einer bestimmten Dichte der Sprites können jedoch keine Einzelheiten mehr erkannt werden, da sich die Sprites durch die XOR-Verknüpfung gegenseitig in ihren Farben beeinflussen, ja sogar auslöschen können.

Um ein möglichst flackerfreies Bild zu erzeugen, was unter BASIC aus Geschwindigkeitsgründen so gut wie unmöglich ist, kann der Befehl CALL &BD19 verwendet werden. Dieser Befehl führt dazu, daß der CPC mit der Ausführung des nächsten Befehls so lange wartet, bis der Schreibstrahl des

Monitors am unteren Ende des Bildes verschwindet, um dann am Bildanfang oben links wieder zu erscheinen. Während dieser Zeit kann der Bildspeicher beschrieben werden, ohne daß das Bild flackert. Aber auch diese Methode bringt nur etwas, wenn wenige Sprites auf dem Bildschirm sind und diese langsam bewegt werden. Eine schnelle flackerfreie Bewegung von Sprites ist nur mittels Maschinensprache möglich. Wer also selbst aufwendige Spiele programmieren will, der wird wohl trotz des schnellen CPC-BASIC nicht um die Assemblerprogrammierung herumkommen.

|SPRITEGEN,@Array%(0,0),nummer

Der zweite Befehl generiert ein Sprite, dessen Farbwerte für jeden Punkt zuvor in einem Integer-Array abgelegt wurden. Der erste Parameter des Befehls ist die Adresse einer Integer-Matrix, die mit DIM Array%(15,15) dimensioniert worden sein muß, während der zweite nur die Nummer des Sprites angibt, das generiert werden soll. Zur Erinnerung sei darauf hingewiesen, daß die Adresse einer Variablen mit dem @ bestimmt wird.

Der Aufruf der SPRITEGEN-Routine muß unbedingt bei Bildschirmmodus 0 geschehen, damit das Sprite richtig generiert wird. Die Integer-Matrix wird so angegeben, daß der Punkt, der bei der Darstellung des Sprites in der linken oberen Ecke liegt, dem Matrixelement 0,0 entspricht. Die erste Dimension der Matrix, also der erste Index eines Matrixelements, entspricht den X-Koordinaten, die zweite den Y-Koordinaten. Jedes Element des Feldes stellt dabei einen Punkt des Sprites dar.

Die generierten Sprites werden mit 128 Byte pro Sprite ab der Adresse &9200 abgelegt. Das bedeutet, daß das Sprite 0 den Speicher &9200–&927F, das Sprite 1 den Speicher &9280–&92FF usw. belegen. Die sechzehn generierten Sprites können dann mit dem Befehl

```
SAVE "NAME",B,&9200,16*128
```

abgespeichert und bei Bedarf wieder in den Speicher geladen werden. Zum Einladen wird das Kommando

```
LOAD "Name"
```

oder

```
LOAD "Name",&9200
```

verwendet.

Soviel zur Beschreibung der beiden neuen Befehle, der folgende Abschnitt behandelt das zugehörige Assemblerlisting. Diejenigen Leser unter Ihnen, die lediglich die beiden Routinen anwenden wollen und sich nicht für die Funktion eines Assemblers und des Programms interessieren, können den folgenden Abschnitt auslassen. Er enthält die Erläuterungen über das Assemblerlisting und die Funktionen des Assemblers. Zum Schluß noch ein kleiner Tip für die Anwendung der Sprites. Die Darstellung der Sprites ist mit Hintergrundfarbe 0 (PAPER 0) am besten, da die Sprites bei einer XOR-Verknüpfung mit 0, also dem Wert der Farbe 0, nicht verändert werden. Bei sehr vielfarbigem Hintergrund ist die Erkennbarkeit der Sprites nicht mehr gegeben, es empfiehlt sich daher, bei der Auswahl des Hintergrundes auf die späteren Sprites Rücksicht zu nehmen. Eine weitere Möglichkeit ist auch die Begrenzung der Sprite-Bewegungen auf einen einfarbigen Hintergrund. Viel Spaß bei der Programmierung von Spielen, vielleicht finden Sie aber auch ein Anwenderprogramm, in dem Sie die Sprites einsetzen können.

8.2 LISTING

```
                      10 *H SPRITES für den CPC464
                      20
BCD1                  30 ERWEIT: EQU  #BCD1
BC1D                  40 DOTPOS: EQU  #BC1D
BC26                  50 NEXTLN: EQU  #BC26
BC2C                  60 INKCOD: EQU  #BC2C
                      70
9200                  80 SPRTAB: EQU  #9200
                      90
9000                 100         ORG  #9000                 ; Anfang des Codes
                     110
9000  010E90         120         LD   BC,TABELL             ; Register für Befehls-
9003  210A90         130         LD   HL,EXTTAB             ; erweiterung laden
9006  CDD1BC         140         CALL ERWEIT                ; Befehle initialisieren
9009  C9             150         RET                        ; zurück zum BASIC
900A                 160 EXTTAB: DEFS 4                     ; Platz für Betriebssys.
900E  1690           170 TABELL: DEFW NAMEN                 ; Sprungtabelle für
9010  C32690         180         JP   SPRITE                ; externe Befehle
9013  C3B990         190         JP   SPRGEN                ; mit den Befehlsnamen
9016  53505249       200 NAMEN:  DEFB "S","P","R","I","T","E"+#80
901C  53505249       210         DEFB "S","P","R","I","T","E","G","E","N"+#80
9025  00             220         DEFB #00
                     230
9026  FE03           240 SPRITE: CP   3                     ; 3 Argumente?
9028  C0             250         RET  NZ                    ; zurück, wenn nicht
9029  DD7E00         260         LD   A,(IX+0)              ; Spritenummer holen
902C  E60F           270         AND  #0F                   ; zw. 0 und 15
902E  DD4603         280         LD   B,(IX+3)              ; Y-Koordinate ins
9031  DD4E02         290         LD   C,(IX+2)              ; BC-Register
9034  DD5605         300         LD   D,(IX+5)              ; X-Koordinate ins
9037  DD5E04         310         LD   E,(IX+4)              ; DE-Register
903A  B7             320         OR   A                     ; Carry löschen
903B  217001         330         LD   HL,400-32             ; Ymax
903E  ED42           340         SBC  HL,BC                 ; Y < Ymax?
9040  D8             350         RET  C                     ; sonst Return
9041  B7             360         OR   A                     ; Carry löschen
9042  214002         370         LD   HL,640-64             ; Xmax
9045  ED52           380         SBC  HL,DE                 ; X < Xmax?
9047  D8             390         RET  C                     ; sonst Return
```

Programm 8.1: Assemblerprogramm zur Implementierung der Sprite-Befehle

```
9048  CB28     400            SRA  B              ; BC-Register auf
904A  CB19     410            RR   C              ; Hälfte reduzieren
904C  CB2A     420            SRA  D              ; DE-Register auf
904E  CB1B     430            RR   E              ; ein Viertel
9050  CB2A     440            SRA  D
9052  CB1B     450            RR   E              ; reduzieren
9054  210F00   460            LD   HL,15          ; Y-Koordinate
9057  09       470            ADD  HL,BC          ; korrigieren
9058  3C       480            INC  A              ; A=A+1 wegen Schleife
9059  E5       490            PUSH HL             ; Y-Koordinate retten
905A  218091   500            LD   HL,SPRTAB-#80  ; mittels HL und BC
905D  018000   510            LD   BC,#80         ; den Tabellenanfang
9060  09       520 TABANF:    ADD  HL,BC          ; bestimmen
9061  3D       530            DEC  A              ; HL := SPRTAB +
9062  20FC     540            JR   NZ,TABANF      ; Spritenr. * 128
9064  E3       550            EX   (SP),HL        ; Y <--> Tabellenanfang
9065  CD1DBC   560            CALL DOTPOS         ; HL=Adresse des Punktes
9068  EB       570            EX   DE,HL          ; DE=Adresse des Punktes
9069  E1       580            POP  HL             ; HL:=Spritematrix
906A  CB41     590            BIT  0,C            ; X-Koordinate ungerade?
906C  3E10     600            LD   A,16           ; Spritehöhe = 16 Zeilen
906E  2016     610            JR   NZ,UNGERA      ; wenn X ungerade,
9070  F5       620 SPZEIL:    PUSH AF             ; wenn X gerade,
9071  D5       630            PUSH DE             ; AF u. DE retten
9072  0608     640            LD   B,8            ; 8 Byte in X-Richtung
9074  1A       650 NXTPIX:    LD   A,(DE)         ; alter Bildschirminhalt
9075  AE       660            XOR  (HL)           ; XOR mit Byte vom Sprite
9076  12       670            LD   (DE),A         ; Byte --> Bildadresse
9077  13       680            INC  DE             ; neue Bildadresse
9078  23       690            INC  HL             ; neue Spriteadresse
9079  10F9     700            DJNZ NXTPIX         ; 8 mal wiederholen
907B  D1       710            POP  DE             ; Adresse holen
907C  EB       720            EX   DE,HL          ; Bildadr.<->Spriteadr.
907D  CD26BC   730            CALL NEXTLN         ; Adresse für nächste
9080  EB       740            EX   DE,HL          ; Spritezeile holen
9081  F1       750            POP  AF             ; Zähler holen
9082  3D       760            DEC  A              ; um 1 erniedrigen
9083  20EB     770            JR   NZ,SPZEIL      ; 16 Zeilen schreiben
9085  C9       780            RET                 ; zum BASIC zurück!
9086           790 UNGERA:                        ; wenn ungerade, dann
9086  F5       800 SPRZEI:    PUSH AF             ; Zähler und Bild-
9087  D5       810            PUSH DE             ; speicheradr. retten
9088  1A       820 PIXEL1:    LD   A,(DE)         ; Bildbyte holen und
9089  4F       830            LD   C,A            ; zwischenspeichern
908A  7E       840            LD   A,(HL)         ; Spritebyte holen
908B  E6AA     850            AND  #AA            ; Bits 7,5,3,1 wählen
908D  0F       860            RRCA                ; nach rechts schieben
908E  A9       870            XOR  C              ; mit gesp. Wert verknü.
908F  12       880            LD   (DE),A         ; neues Byte --> Bild
9090  13       890            INC  DE             ; nächste Bildadr.
9091  0607     900            LD   B,7            ; 7 Bytes = 14 Pixels
9093  7E       910 LOOP:      LD   A,(HL)         ; Byte nochmal holen
9094  E655     920            AND  #55            ; Bits 6,4,2,0 wählen
9096  07       930            RLCA                ; nach links schieben
9097  4F       940            LD   C,A            ; zwischenspeichern
9098  23       950            INC  HL             ; nächstes Spritebyte
9099  7E       960            LD   A,(HL)         ; holen und Bits
909A  E6AA     970            AND  #AA            ; 7,5,3,1 wählen
909C  0F       980            RRCA                ; nach rechts schieben
909D  B1       990            OR   C              ; mit gesp. Wert verknü.
909E  4F       1000           LD   C,A            ; und speichern
909F  1A       1010           LD   A,(DE)         ; Bildbyte holen
90A0  A9       1020           XOR  C              ; mit gesp. Wert verknü.
90A1  12       1030           LD   (DE),A         ; neues Byte --> Bild
90A2  13       1040           INC  DE             ; nächstes Bildbyte
90A3  10EE     1050           DJNZ LOOP           ; 7mal wdh.
90A5  7E       1060 PIX_16:   LD   A,(HL)         ; Spritebyte holen
90A6  E655     1070           AND  #55            ; Bits 6,4,2,0 wählen
90A8  07       1080           RLCA                ; nach links schieben
90A9  4F       1090           LD   C,A            ; zwischenspeichern
```

Programm 8.1: Assemblerprogramm zur Implementierung der Sprite-Befehle (Forts.)

```
90AA  1A      1100          LD   A,(DE)          ; Bildbyte holen
90AB  A9      1110          XOR  C               ; mit gesp. Wert verknü.
90AC  12      1120          LD   (DE),A          ; neues Byte --> Bild
90AD  23      1130          INC  HL              ; nächstes Spritebyte
90AE  D1      1140          POP  DE              ; Bildadresse holen
90AF  EB      1150          EX   DE,HL           ; Bildadr.<->Spriteadr.
90B0  CD26BC  1160          CALL NEXTLN          ; nächste Bildz. best.
90B3  EB      1170          EX   DE,HL           ; Spriteadr.<->Bildadr.
90B4  F1      1180          POP  AF              ; Zähler holen
90B5  3D      1190          DEC  A               ; um 1 erniedrigen
90B6  20CE    1200          JR   NZ,SPRZEI       ; 16mal wdh.
90B8  C9      1210          RET                  ; zum BASIC
              1220
90B9  FE02    1230 SPRGEN:  CP   2               ; 2 Argumente?
90BB  C0      1240          RET  NZ              ; zurück, wenn nicht
90BC  DD7E00  1250          LD   A,(IX+0)        ; Spritenummer
90BF  E60F    1260          AND  #0F             ; Nr. zw. 0 und 15
90C1  47      1270          LD   B,A             ; --> B-Register
90C2  04      1280          INC  B               ; mind. 1 Durchlauf
90C3  218091  1290          LD   HL,SPRTAB-#80   ; aus Spritetabellen-
90C6  118000  1300          LD   DE,#80          ; anfang und Spritegröße
90C9  19      1310 TABBEG:  ADD  HL,DE           ; den Anfang des Sprites
90CA  10FD    1320          DJNZ TABBEG          ; berechnen
90CC  DD5603  1330          LD   D,(IX+3)        ; Adresse der Matrix
90CF  DD5E02  1340          LD   E,(IX+2)        ; ins DE-Register
90D2  EB      1350          EX   DE,HL           ; Matrix und Tabellen-
              1360 ;                             adresse vertauschen
90D3  0680    1370          LD   B,128           ; 128 Byte pro Sprite
90D5  7E      1380 INKLOP:  LD   A,(HL)          ; Matrixink holen
90D6  CD2CBC  1390          CALL INKCOD          ; codieren und
90D9  E6AA    1400          AND  #AA             ; Bits 7,5,3,1 wählen
90DB  4F      1410          LD   C,A             ; zwischenspeichern
90DC  23      1420          INC  HL              ; nächsten Matrixwert
90DD  23      1430          INC  HL              ; wählen und
90DE  7E      1440          LD   A,(HL)          ; Matrixink holen
90DF  CD2CBC  1450          CALL INKCOD          ; Ink codieren und
90E2  E655    1460          AND  #55             ; Bits 6,4,2,0 wählen
90E4  B1      1470          OR   C               ; mit vorigem Wert ODER
90E5  12      1480          LD   (DE),A          ; verknüpfen,abspeichern
90E6  13      1490          INC  DE              ; Zieladresse + 1
90E7  23      1500          INC  HL              ; Quelladresse + 2
90E8  23      1510          INC  HL              ; bestimmen
90E9  10EA    1520          DJNZ INKLOP          ; 128mal wiederholen
90EB  C9      1530          RET                  ; zurück zum BASIC

Pass 2 errors: 00

Table used:   270  from   651
```

Programm 8.1: Assemblerprogramm zur Implementierung der Sprite-Befehle (Forts.)

```
65000 'BASIC-Loader für die Spriteroutinen SPRITE und SPRITEGEN
65010 'Das RETURN in Zeile 65180 ist für die Einbindung als
65020 'Unterprogramm gedacht.
65030 RESTORE 65500
65040 IF HIMEM>36863 THEN MEMORY &8FFF
65050 IF PEEK(&90EB)=&C9 THEN RETURN
65060 zeile=0
65070 FOR n%=&9000 TO &90EF STEP 16
65080     zeile=zeile+1
65090     summe%=0
65100     FOR i%=0 TO 15
65110         READ byte$
65120         byte%=VAL("&"+byte$)
65130         summe%=summe%+byte%
```

Programm 8.2: BASIC-Lader

```
65140           POKE n%+i%,byte%
65150       NEXT
65160       READ check$
65170       check%=VAL("&"+check$)
65180       IF check%=summe%THEN 65210
65190          PRINT" Fehler beim Laden der Spriteroutinen !!!"
65200          PRINT" Fehler in der";zeile;". DATA-Zeile !":END
65210 NEXT
65220 CALL &9000
65230 RETURN
65500 'DATA-Zeilen für die Spriteroutinen
65501 DATA 01,0E,90,21,0A,90,CD,D1,BC,C9,00,00,00,00,16,90,0523
65502 DATA C3,26,90,C3,B9,90,53,50,52,49,54,C5,53,50,52,49,071A
65503 DATA 54,45,47,45,CE,00,FE,03,C0,DD,7E,00,E6,0F,DD,46,0727
65504 DATA 03,DD,4E,02,DD,56,05,DD,5E,04,B7,21,70,01,ED,42,061F
65505 DATA D8,B7,21,40,02,ED,52,D8,CB,28,CB,19,CB,2A,CB,1B,07BB
65506 DATA CB,2A,CB,1B,21,0F,00,09,3C,E5,21,80,91,01,80,00,04E8
65507 DATA 09,3D,20,FC,E3,CD,1D,BC,EB,E1,CB,41,3E,10,20,16,0747
65508 DATA F5,D5,06,08,1A,AE,12,13,23,10,F9,D1,EB,CD,26,BC,075C
65509 DATA EB,F1,3D,20,EB,C9,F5,D5,1A,4F,7E,E6,AA,0F,A9,12,08F8
65510 DATA 13,06,07,7E,E6,55,07,4F,23,7E,E6,AA,0F,B1,4F,1A,0589
65511 DATA A9,12,13,10,EE,7E,E6,55,07,4F,1A,A9,12,23,D1,EB,068F
65512 DATA CD,26,BC,EB,F1,3D,20,CE,C9,FE,02,C0,DD,7E,00,E6,0980
65513 DATA 0F,47,04,21,80,91,11,80,00,19,10,FD,DD,56,03,DD,0556
65514 DATA 5E,02,EB,06,80,7E,CD,2C,BC,E6,AA,4F,23,23,7E,CD,0774
65515 DATA 2C,BC,E6,55,B1,12,13,23,23,10,EA,C9,00,00,00,00,0502
```

Programm 8.2: BASIC-Lader (Forts.)

8.3 PROGRAMMBESCHREIBUNG

Bei der folgenden Beschreibung des Assemblerlistings soll in Reihenfolge der Zeilennummern des Assemblers vorgegangen werden. In der ersten Assemblerzeile steht lediglich der Name des Programms. Die Anweisung *H ist eine sogenannte Assemblerdirektive und damit nur für den Assembler als Befehl gedacht, der nichts mit dem Programm zu tun hat. Diese Direktiven sind bei den meisten Assemblern unterschiedlich, haben Sie also einen anderen Assembler, so lassen Sie diese Anweisung am besten weg.

Die Zeilen 30 bis 80 stellen die Definition wichtiger Adressen dar. Durch die Benutzung dieser sogenannten LABELS wird ein Assemblerprogramm übersichtlicher und kann leichter auf andere Speicheradressen umgeschrieben werden, da die Definition am Anfang des Programms erfolgt. Diese Namen stellen Konstanten dar, denen ein Wert mittels der EQU-Anweisung zugewiesen wird. Hätte man diese Labels nicht, so müßte man bei einer Programmänderung alle Stellen mit dem entsprechenden Wert ändern. In unserem Fall muß man dies nur bei der EQU-Anweisung.

Die Labels können beim vorliegenden Assembler nur sechs Zeichen lang sein, deshalb nimmt man meist mehr oder weniger sinnvolle Abkürzungen für die entsprechenden Adressen, so z. B. ERWEIT für den Sprung, der die Befehlserweiterung in das Betriebssystem einbindet. Der entsprechende CALL oder JP geht dann an die Adresse &BCD1.

Der Assembler verlangt übrigens als Kennzeichnung für eine Hexzahl den Vorsatz #, während das BASIC den Vorsatz & bzw. &H verwendet. Der Sprung an die Adresse &BCD1, also ERWEIT, benötigt im BC-Register die Adresse der Sprungtabelle der Befehle. Im HL-Register muß die Adresse eines vier Byte langen unbelegten Speicherbereichs übergeben werden.

Der Sprung nach &BC1D, also DOTPOS, führt eine Berechnung der Bildspeicheradresse eines Bildpunktes aus. Dieser Bildpunkt wird der Routine als Koordinatenpaar im DE- und HL-Register angegeben. Die Koordinaten werden nicht wie die üblichen Grafikkoordinaten angegeben, sondern als Koordinaten eines einzelnen adressierbaren Bildpunktes. Sie sind je nach Modus unterschiedlich. So liegt die rechte obere Ecke im MODE 0 nicht mehr bei der Y-Koordinate 398 oder 399 und der X-Koordinate im Bereich von 632 bis 639, sondern bei (159,199), da die Auflösung 160*200 Bildpunkte beträgt. Das DE-Register gibt die X-Koordinate an und das HL-Register die Y-Koordinate. Die Routine liefert die Bildspeicheradresse des Punktes im HL-Register und die Maske für diesen Punkt im C-Register zurück.

Der Sprung nach &BC26, also NEXTLN, berechnet die Adresse des Bytes in der nächsten Zeile des Bildes. Dazu ist der Routine die Speicheradresse eines Bytes im HL-Register zu übergeben. Die Routine berechnet daraus die Adresse des Bytes, das eine Punktreihe unter dem übergebenen liegt. Die errechnete Adresse wird im HL-Register zurückgegeben.

Die Routine bei &BC2C, also INKCOD, codiert die im Akku übergebene Inknummer zu der dieser Inknummer entsprechenden Bitfolge in einem Byte. Das Ergebnis wird im Akku zurückgegeben.

Die Festlegung SPRTAB: EQU #9200 gibt an, daß die Tabelle der Sprites bei &9200 beginnen soll. Die nachfolgende ORG-Anweisung befiehlt dem Assembler, daß der erzeugte Maschinencode ab der Adresse &9000 abgelegt werden soll. Danach beginnt das eigentliche Programm mit der Einbindung der beiden Sprite-Befehle in den Zeilen 120 bis 150, die dazu notwendigen Daten stehen in den Zeilen 160 bis 220. Anschließend kehrt das Programm wieder in das BASIC zurück.

Nun zu der Beschreibung der beiden Sprite-Routinen: Die erste Routine (SPRITE) wird von Zeile 180 mit JP SPRITE angesprungen. Als erstes vergleicht die Routine den Akkuinhalt mit drei. Das geschieht zur Feststellung, ob auch drei Parameter der Routine übergeben wurden. Ist dies nicht der Fall, so erfolgt sofort eine Rückkehr ins BASIC mit RET NZ.

Waren tatsächlich drei Parameter vorhanden, so wird zuerst die Sprite-Nummer geholt. Dies wird mit der Anweisung LD A,(IX+0) erledigt, die das niederwertige Byte des letzten Parameters in den Akku lädt. Die anschließende

UND-Verknüpfung mit &0F sorgt dafür, daß die Sprite-Nummer zwischen 0 und 15 liegt.

Darauf wird die Y-Koordinate in das BC-Register und die X-Koordinate in das DE-Register geladen. Um nun die Koordinaten auf ihre Gültigkeit zu prüfen, werden sie jeweils bei gelöschtem Übertragsflag (Carry) vom Maximalwert subtrahiert. Tritt dabei ein Übertrag auf, ist also die Koordinate größer als der Maximalwert, so kehrt die Routine ins BASIC zurück.

Lagen die Koordinaten im gültigen Bereich, wird die Y-Koordinate auf die Hälfte und die X-Koordinate auf ein Viertel reduziert, damit der Routine DOTPOS die echten Pixelkoordinaten übergeben werden können. Nach dieser Korrektur, die mit einer Schiebeoperation durchgeführt wird, wird der Y-Wert noch um 15 erhöht. Diese Addition dient nur zum vereinfachten Einschreiben des Sprites, so daß mit dem oberen linken Sprite-Punkt begonnen werden kann.

Nun wird die Anfangsadresse des Sprites berechnet. Es wird in Zeile 480 zuerst der Akkuinhalt, also die Sprite-Nummer, um 1 erhöht, man sagt auch: der Akku wird inkrementiert. Dies ist notwendig, damit die Routine zur Berechnung der Anfangsadresse mindestens einmal durchlaufen wird und bei Sprite-Nummer 0 nicht etwa 255mal. Die Y-Koordinate muß, da sie sich seit der Addition mit 15 im HL-Register befindet, zwischengespeichert werden, weil das HL-Register für die Berechnung benötigt wird. Das Register wird nun mit der Anfangsadresse der Sprite-Tabelle minus der Speicherlänge eines Sprites geladen, da die Schleife mindestens einmal durchlaufen wird. Das BC-Register wird mit der Länge eines Sprite-Bereichs, also 128 Byte (Hex 80), geladen.

Dieser Distanzwert wird nun mittels einer Schleife so oft zur Adresse in HL addiert, bis die Adresse des Sprites erreicht ist. Jetzt wird die Schleife verlassen und die Y-Koordinate mit dem Tabellenanfang ausgetauscht. Dadurch stehen jetzt in den Registern DE und HL die Koordinaten für den Aufruf von DOTPOS zur Verfügung.

Die DOTPOS-Routine liefert in HL die Bildspeicheradresse des Punktes zurück. Diese Adresse wird aber in DE benötigt, und deshalb wird der Inhalt von DE und HL ausgetauscht (EX DE,HL) und die Adresse des Sprites nach HL geladen.

Die Routine hat aber auch noch einen Hinweis auf die Lage des Punktes gegeben. Der Befehl BIT 0,C testet das niederwertigste Bit im Register C und gibt dadurch einen Hinweis, ob der erste Punkt mit dem Anfang eines Bytes übereinstimmt oder nicht. In letzterem Fall wird dann auf den Abschnitt für den ungeraden Sprite-Anfang verzweigt. Doch davor wird noch der Akku mit der Anzahl der Bildzeilen des Sprites geladen. Dies hätte man auch, ebenso wie

die Berechnung des Punktes, in jedem Abschnitt für sich durchführen lassen können. Aber unsere Methode spart Speicherplatz.

Die nun folgende Beschreibung ist in die beiden Abschnitte gerader und ungerader Sprite-Anfang unterteilt. Zuerst, wie im Programm, zum geraden Abschnitt. Die Einschreibe-Routine muß, damit später wieder darauf zugegriffen werden kann, die Inhalte der Registerpaare AF und DE zwischenspeichern. Dann wird das Register B als Bytezähler in X-Richtung mit 8 geladen. Nun wird der Inhalt des Bildschirms in den Akku geladen, die Adresse dazu steht im DE-Register. Dieser alte Bildschirminhalt wird nun mit dem Sprite EXKLUSIV ODER-verknüpft, dessen Adresse in HL steht.

Der so neugewonnene Bildwert muß nur noch in den Bildspeicher zurückgeschrieben werden. Im folgenden werden die Bildadresse und die Sprite-Adresse inkrementiert, damit das nächste Byte so behandelt werden kann. Der Befehl DJNZ NXTPIX erniedrigt den Bytezähler B um 1 und vollführt – wenn noch nicht 0 erreicht ist – einen Sprung zu NXTPIX, um das nächste Byte dieser Zeile des Sprites einzuschreiben. Das Ganze wiederholt sich achtmal, bis der Inhalt von B 0 geworden ist.

Jetzt wird die zwischengespeicherte Bildspeicheradresse zurückgeholt, DE und HL werden vertauscht, die Adresse in HL wird mittels der CPC-Routine NEXTLN auf die Bildzeile darunter korrigiert, und DE wird wieder mit HL vertauscht. Anschließend wird der Zeilenzähler im Akku dekrementiert und – solange er nicht 0 ist – zu SPZEIL gesprungen, um die nächste Zeile des Sprites auf den Bildschirm zu bringen. Wurde dies sechzehnmal wiederholt, so steht das Sprite am Bildschirm, und die Routine kehrt ins BASIC zurück.

Der Abschnitt für die ungeraden Pixels arbeitet etwas komplizierter, da aus je einem Byte des Sprites und des Bildschirms jeweils ein Punkt selektiert wird, und die beiden Punkte verknüpft werden müssen. Der Anfang der Routine ist der gleiche wie bei geraden Pixels, der Zeilenzähler und die Bildschirmadresse werden zwischengespeichert. Da das erste und das letzte Pixel einer Sprite-Zeile jetzt anders behandelt werden müssen, kann nicht eine einfache Zählschleife die acht Bytes pro Sprite-Zeile einschreiben.

Die Einschreib-Routine für die ungeraden X-Koordinaten ist deshalb in drei Abschnitte unterteilt: PIXEL1, LOOP und PIX_16. Der erste Abschnitt schreibt – wie der Name schon andeutet – das erste Pixel einer Zeile in den Bildspeicher. Dazu wird der alte Bildinhalt wieder in den Akku geladen und im Register C zwischengespeichert.

Jetzt wird das erste Byte dieser Sprite-Zeile in den Akku geholt, und mit einer UND-Verknüpfung werden die Bits 7, 5, 3 und 1 selektiert. Diese Bits stellen das erste Pixel des Sprites dar. Das Pixel muß von seiner linken Lage im Byte

an die rechte gebracht werden. Dies geschieht mit der Verschiebung durch RRCA. RRCA rotiert den Akkuinhalt nach rechts, d. h. jetzt liegen die selektierten Bits an den Positionen 6, 4, 2 und 0. Dieses ausgewählte und verschobene Pixel wird nun mit dem in C gespeicherten alten Bildinhalt XOR-verknüpft und zuletzt wieder in den Bildspeicher eingeschrieben.

Für die folgenden vierzehn Pixels (sieben Bytes) kann eine Schleife verwendet werden. Zum Eintritt in diese Schleife wird die Bildadresse inkrementiert und das Register B als Bytezähler mit 7 geladen. Das nur zur Hälfte gebrauchte erste Byte der Sprite-Zeile wird nochmals in den Akku geholt. Jetzt werden aber die Bits 6, 4, 2 und 0 ausgewählt und nach links verschoben. Dies ist der erste Teil der beiden gewünschten Pixels. Deshalb wird dieser Wert im Register C zwischengespeichert. Der zweite Teil wird aus dem nächsten Byte des Sprites gewonnen. Die dazu erforderliche Selektion holt die Bits 7, 5, 3 und 1 und schiebt sie auf die Positionen 6, 4, 2 und 0. Der so erhaltene Wert wird mit dem ersten Teil der Pixels zusammengeführt (ODER-Verknüpfung).

Als nächstes kann das Bildbyte geholt, mit dem erhaltenen Wert XOR-verknüpft und wieder zurück in den Bildspeicher gebracht werden.

Zum Schluß dieser Schleife wird das nächste Bildbyte adressiert, der Bytezähler erniedrigt und bei einem Wert ungleich 0 zum Anfang der Schleife gesprungen.

Ansonsten wird der letzte Bildpunkt der Sprite-Zeile behandelt. Dazu wird das zuletzt angesprochene Sprite-Byte noch mal geholt, die Bits 6, 4, 2 und 0 werden ausgewählt und auf die Positionen 7,5,3 und 1 geschoben. Dieser Wert wird zwischengespeichert, und das entsprechende Byte des Bildspeichers wird geholt. Es wird mit dem gespeicherten Wert XOR-verknüpft und in den Bildspeicher zurückgeschrieben. Dann wird das nächste Byte des Sprites adressiert, die Bildadresse geholt, DE mit HL vertauscht, die neue Bildadresse ermittelt und schließlich DE wieder mit HL vertauscht.

Als letztes wird der Zeilenzähler geholt, um 1 erniedrigt und – wenn das Ergebnis größer als 0 ist – die nächste Sprite-Zeile eingeschrieben, sonst erfolgt ein Rücksprung zum BASIC.

Die Routine SPRITEGEN wird vom zweiten Eintrag der Befehlserweiterung mit JP SPRGEN angesprungen. Zuerst erfolgt wieder eine Überprüfung der Parameterzahl. Ist sie ungleich 2, so kehrt die Routine ins BASIC zurück. Ansonsten wird die Sprite-Nummer geholt und auf einen Wert zwischen 0 und 15 gebracht. Das Register B wird mit der Sprite-Nummer geladen und inkrementiert, um, wie bei der SPRITE-Routine, die Berechnung des Anfangs der Sprite-Tabelle für dieses Sprite ordnungsgemäß durchführen zu können. Die Berechnung der Adresse des Sprites erfolgt wie bei der SPRITE-Routine.

Nach der Berechnung wird die Adresse der Integer-Matrix ins DE-Register geladen, und DE und HL werden vertauscht. Jetzt wird das B-Register mit 128 – also der Anzahl der Bytes pro Sprite – geladen, und die Schleife zur Umsetzung der Matrix in die Tabelle wird betreten. Dazu wird der Akku mit dem Wert der Matrix geladen.

Die CPC-Routine INKCOD codiert den Inkwert in einen entsprechenden Bytewert um. Von diesem werden die Bits 7, 5, 3 und 1 ausgewählt, und der erhaltene Wert wird zwischengespeichert. Das HL-Register wird um 2 erhöht, um den nächsten Matrixeintrag laden zu können. Dieser wird wieder durch INKCOD codiert. Die Bits 6, 4, 2 und 0 werden selektiert, mit dem zwischengespeicherten Wert ODER-verknüpft und als neuer Bytewert des Sprites eingetragen. Dann wird das nächste Sprite-Byte und der nächste Matrixeintrag adressiert. Das Ganze wird 128mal wiederholt und damit die ganze Integer-Matrix in ein Sprite umgewandelt. Zum Schluß erfolgt der obligatorische Rücksprung ins BASIC.



Kapitel 9
Hardcopy

Gleich zu Anfang soll klargestellt werden, was eigentlich eine Hardcopy ist, denn die Computer-Neulinge unter Ihnen werden wahrscheinlich nichts mit dem Begriff anfangen können. Eine Hardcopy ist eine – möglichst maßstabsgetreue – Abbildung des Bildschirminhalts auf dem Drucker oder gar einem Plotter. Gute Hardcopy-Programme können sogar auf speziellen Druckern eine auch farbgetreue Abbildung liefern. Da nicht davon ausgegangen werden kann, daß jeder von Ihnen einen Farbdrucker hat, und alle Farbdrucker unterschiedlich programmiert werden müssen, soll im folgenden ein Hardcopy-Programm für einfarbige Hardcopys vorgestellt werden.

Das Hardcopy-Programm wird als Befehlserweiterung in das Betriebssystem eingebunden und kann von BASIC mit |KOPIE aufgerufen werden.

Die Routine kann durch den abgedruckten BASIC-Lader eingebunden werden. Dazu ist der Lader abzutippen und mit

```
GOSUB 64000
```

aufzurufen. Der Lader schreibt dann die Routine in den Speicher und initialisiert sie mit einem CALL &A200, bevor ein RETURN erfolgt. Wenn im abgetippten Programm fehlerhafte Bytes sind, so erkennt, wenn sie sich nicht gegenseitig aufheben, dies die Routine und gibt eine Warnung aus, bevor sie das POKEn aufhört.

Eine andere Möglichkeit ist das Abtippen des Assemblerlistings in einen geeigneten Assembler. Das hat zudem den Vorteil, daß die Anfangsadresse mit der ORG-Anweisung verändert werden kann. Das Programm muß dann nach dem Eintippen assembliert werden. Nach der Assemblierung muß die Routine mit einem

```
CALL &A200
```

oder

```
CALL ORG-Adresse
```

initialisiert werden, bevor der Befehl zugänglich wird.

Das assemblierte Programm kann, ebenso wie das durch den BASIC-Lader erzeugte, mit dem Befehl

SAVE "HARDCOPY",B,&A200,&140

gespeichert und mit dem Befehl

MEMORY &A1FF : LOAD "HARDCOPY" : CALL &A200

bei Bedarf eingeladen und initialisiert werden. Das hat den Vorteil, daß der Lader keinen Platz im Programm einnimmt bzw. die Routine nicht immer nach einem Reset vor ihrer Anwendung assembliert werden muß.

Nachdem der Befehl initialisiert ist, kann er durch die Anweisung |KOPIE benutzt werden. Den senkrechten Strich vor dem Befehl erreicht man durch Drücken einer SHIFT-Taste und der Taste rechts nebem dem P.

```
Pass 1 errors: 00

BCD1                10 ERWEIT:  EQU  #BCD1
BB1B                20 GETKEY:  EQU  #BB1B
BBBA                30 GRINIT:  EQU  #BBBA
BBC0                40 MOVE:    EQU  #BBC0
BBC6                50 XY_POS:  EQU  #BBC6
BBC9                60 SETORG:  EQU  #BBC9
BBCC                70 GETORG:  EQU  #BBCC
BBCF                80 SETX12:  EQU  #BBCF
BBD2                90 SETY12:  EQU  #BBD2
BBD5               100 GETX12:  EQU  #BBD5
BBD8               110 GETY12:  EQU  #BBD8
BBDE               120 SGRPEN:  EQU  #BBDE
BBE1               130 GGRPEN:  EQU  #BBE1
BBE4               140 SGRPAP:  EQU  #BBE4
BBE7               150 GGRPAP:  EQU  #BBE7
BBF0               160 TEST:    EQU  #BBF0
BC11               170 GTMODE:  EQU  #BC11
BD2E               180 TESTPR:  EQU  #BD2E
BD2B               190 SENDPR:  EQU  #BD2B
                   200
00FC               210 BREAK:   EQU  #FC
001B               220 ESC:     EQU  #1B
000D               230 CR:      EQU  #0D
000A               240 LF:      EQU  #0A
                   250
                   260
A200               270          ORG  #A200
                   280
A200  0109A2       290          LD   BC,TABELLE          ; Tabellenadr. laden
A203  2114A2       300          LD   HL,EXTTAB           ; Adr. 4 reserv. Byte
A206  C3D1BC       310          JP   ERWEIT              ; Befehl einbinden
A209  0EA2         320 TABELL:  DEFW NAME                ; Adr. des Namens
A20B  C318A2       330          JP   KOPIE               ; Sprung zur Hardcopy
A20E  4B4F5049     340 NAME:    DEFB "K","O","P","I","E"+#80,#00
A214               350 EXTTAB:  DEFS 4                   ; Platz für System
                   360
A218  CDC9A2       370 KOPIE:   CALL GRSTO               ; Grafikwerte speichern
A21B  CD11BC       380          CALL GTMODE              ; Aktuellen Modus holen
A21E  3C           390          INC  A                   ; A = MODE + 1
A21F  47           400          LD   B,A                 ; Zählregister laden
A220  3E08         410          LD   A,8                 ; A = 2 * max. Step
```

Programm 9.1: Assemblerlisting der Hardcopy-Routine

```
A222  0F        420 ST_LP:   RRCA              ; A = A / 2
A223  10FD      430          DJNZ ST_LP        ; sooft wie MODE + 1
A225  3274A2    440          LD   (STEP),A     ; Stepwert speichern
A228  CDBABB    450          CALL GRINIT       ; Reset des Grafiksys.
A22B  3E1B      460          LD   A,ESC        ; Drucker auf 7/72"
A22D  CD9DA2    470          CALL PRCHR        ; Zeilenvorschub
A230  3E31      480          LD   A,"1"        ; einstellen
A232  CD9DA2    490          CALL PRCHR
A235  218F01    500          LD   HL,399       ; oben anfangen (Y:=399)
A238            510 NXTLIN:                    ; Zeile ausgeben
A238  CDB0A2    520 GRAINI:  CALL PRCRLF       ; Zeilenvorschub
A23B  3E1B      530          LD   A,ESC        ; Drucker für die
A23D  CD9DA2    540          CALL PRCHR        ; Ausgabe von
A240  3E4C      550          LD   A,"L"        ; Grafik mit
A242  CD9DA2    560          CALL PRCHR        ; 768 Punkten
A245  AF        570          XOR  A            ; = 0 Punkte +
A246  CD9DA2    580          CALL PRCHR
A249  3E03      590          LD   A,3          ; + 3*256 Punkte
A24B  CD9DA2    600          CALL PRCHR        ; vorbereiten
A24E  0680      610          LD   B,128        ; 128 * CHR$(0)
A250  AF        620 LOOP:    XOR  A            ; ausgeben
A251  CD9DA2    630          CALL PRCHR        ; --> noch 640 Punkte
A254  10FA      640          DJNZ LOOP         ; übrig
A256  110000    650          LD   DE,0         ; links anfangen (X:=0)
A259  E5        660 XLOOP:   PUSH HL           ; Y sichern
A25A  0607      670          LD   B,7          ; Zähler für 7 Punkte
A25C  0E00      680          LD   C,0          ; CHR$ ist erst 0
A25E  CDBBA2    700 DTLOOP:  CALL TESTXY       ; Punkt testen
A261  CB11      710          RL   C            ; Carry in LSBit rot.
A263  2B        730          DEC  HL           ; Y um einen Pixel
A264  2B        740          DEC  HL           ; = 2 Punkte erniedrigen
A265  10F7      750          DJNZ DTLOOP       ; nächster der 7 Punkte
A267  79        760          LD   A,C          ; CHR$ --> A
A268  E1        770          POP  HL           ; Y holen
A269  E5        780          PUSH HL           ; und wieder sichern
A26A  010800    790          LD   BC,8         ; Sind es die
A26D  ED42      800          SBC  HL,BC        ; 4 letzten Dotreihen?
A26F  3002      810          JR   NC,NOLAST    ; ja? Dann die obersten
A271  E678      820          AND  %1111000     ; 4 Druckbits selekt.
A273  0600      830 NOLAST:  LD   B,#00        ; Stepwert laden
A274            840 STEP:    EQU  $-1          ; Argument v. LD B,..
A275  CD9DA2    850 STEPLP:  CALL PRCHR        ; CHR$ ausgeben
A278  13        860          INC  DE           ; X := X + 1
A279  10FA      870          DJNZ STEPLP       ; sooft wie Stepzahl
A27B  B7        880          OR   A            ; Carry löschen
A27C  217F02    890          LD   HL,639       ; Max. X-Wert laden
A27F  ED52      900          SBC  HL,DE        ; x kleiner 639?
A281  E1        910          POP  HL           ; (Y holen)
A282  F259A2    920          JP   P,XLOOP      ; dann nächster X-Wert,
A285  B7        930          OR   A            ; sonst Carry löschen
A286  010E00    940          LD   BC,14        ; BC:=14 (= 7 Pixel)
A289  ED42      950          SBC  HL,BC        ; Y:=Y-14, wenn Y>0
A28B  30AB      960          JR   NC,NXTLIN    ; dann nächste Zeile
A28D  3E1B      970 ENDE:    LD   A,ESC        ; sonst Zeilenvorschub
A28F  CD9DA2    980          CALL PRCHR        ; auf 1/6 inch
A292  3E32      990          LD   A,"2"        ; zurückstellen
A294  CD9DA2   1000          CALL PRCHR
A297  CDB0A2   1010          CALL PRCRLF       ; und Zeilenvorschub
A29A  C303A3   1020          JP   GRREST       ; Grafikwerte setzen
               1030
A29D  4F       1040 PRCHR:   LD   C,A          ; Zeichen retten
A29E  CD1BBB   1050 WAITPR:  CALL GETKEY       ; Taste holen
A2A1  FEFC     1060          CP   BREAK        ; ESC-Taste testen
A2A3  CA03A3   1070          JP   Z,GRREST     ; Grafikwerte setzen
A2A6  CD2EBD   1080 NOBRK:   CALL TESTPR       ; sonst wieder Druck-
A2A9  38F3     1090          JR   C,WAITPR     ; bereitschaft testen
A2AB  79       1100          LD   A,C          ; wenn bereit, Zeichen
A2AC  CD2BBD   1110          CALL SENDPR       ; holen und ausgeben
A2AF  C9       1120          RET               ; und Rückkehr
               1130
```

Programm 9.1: Assemblerlisting der Hardcopy-Routine (Forts.)

```
A2B0  3E0D      1140 PRCRLF: LD   A,CR            ; CR und
A2B2  CD9DA2    1150         CALL PRCHR
A2B5  3E0A      1160         LD   A,LF            ; LF ausgeben
A2B7  CD9DA2    1170         CALL PRCHR           ; = Zeilenvorschub
A2BA  C9        1180         RET                  ; Rückkehr
                1190
A2BB  C5        1200 TESTXY: PUSH BC              ; BC, DE und HL
A2BC  D5        1210         PUSH DE              ; auf den Stack
A2BD  E5        1220         PUSH HL              ; retten
A2BE  CDF0BB    1230         CALL TEST            ; Punkt (DE,HL) testen
A2C1  B7        1240         OR   A               ; Carry löschen u. A=0?
A2C2  2801      1250         JR   Z,NOTSET        ; A=0, dann nichts tun
A2C4  37        1260         SCF                  ; sonst Carry setzen
A2C5  E1        1270 NOTSET: POP  HL              ; HL, DE und BC
A2C6  D1        1280         POP  DE              ; wieder vom Stack
A2C7  C1        1290         POP  BC              ; holen
A2C8  C9        1300         RET                  ; und Rückkehr
                1310
A2C9  CDC6BB    1320 GRSTO:  CALL XY_POS          ; Cursorpos. abfragen
A2CC  ED5322A3  1330         LD   (X_POS),DE      ; X- und Y-Koordinate
A2D0  2225A3    1340         LD   (Y_POS),HL      ; speichern
A2D3  CDCCBB    1350         CALL GETORG          ; Origin abfragen
A2D6  ED5307A3  1360         LD   (ORG_X),DE      ; X- und Y-Koordinate
A2DA  220AA3    1370         LD   (ORG_Y),HL      ; speichern
A2DD  CDD5BB    1380         CALL GETX12          ; Grafikfensterbreite
A2E0  2210A3    1390         LD   (WIN_X1),HL     ; abfragen und X1 und
A2E3  ED5313A3  1400         LD   (WIN_X2),DE     ; X2 speichern
A2E7  CDD8BB    1410         CALL GETY12          ; Grafikfensterhöhe
A2EA  2219A3    1420         LD   (WIN_Y1),HL     ; abfragen und Y1 und
A2ED  ED531CA3  1430         LD   (WIN_Y2),DE     ; Y2 speichern
A2F1  CDE1BB    1440         CALL GGRPEN          ; Grafikpen holen
A2F4  322BA3    1450         LD   (GRAPEN),A      ; Grafikstift speichern
A2F7  CDE7BB    1460         CALL GGRPAP          ; Grafikpaper holen
A2FA  3230A3    1470         LD   (GRAPAP),A      ; Grafikpaper speichern
A2FD  E1        1480         POP  HL              ; Returnadresse holen
A2FE  ED7304A3  1490         LD   (STPTR),SP      ; Stackpointer speichern
A302  E9        1500         JP   (HL)            ; Return durchführen
                1510
A303  31F2BF    1520 GRREST: LD   SP,#BFF2        ; Stackpointer laden
A304            1530 STPTR:  EQU  $-2             ; Argument v. LD SP,....
A306  110000    1540         LD   DE,#0000        ; X-Origin laden
A307            1550 ORG_X:  EQU  $-2             ; Argument v. LD DE,....
A309  210000    1560         LD   HL,#0000        ; Y-Origin laden
A30A            1570 ORG_Y:  EQU  $-2             ; Argument v. LD HL,....
A30C  CDC9BB    1580         CALL SETORG          ; Origin setzen
A30F  110000    1590         LD   DE,#0000        ; Windowbreite1 laden
A310            1600 WIN_X1: EQU  $-2             ; Argument v. LD DE,....
A312  210000    1610         LD   HL,#0000        ; Windowbreite2 laden
A313            1620 WIN_X2: EQU  $-2             ; Argument v. LD HL,....
A315  CDCFBB    1630         CALL SETX12          ; Windowbreite setzen
A318  110000    1640         LD   DE,#0000        ; Windowhöhe1 laden
A319            1650 WIN_Y1: EQU  $-2             ; Argument v. LD DE,....
A31B  210000    1660         LD   HL,#0000        ; Windowhöhe2 laden
A31C            1670 WIN_Y2: EQU  $-2             ; Argument v. LD HL,....
A31E  CDD2BB    1680         CALL SETY12          ; Windowhöhe setzen
A321  110000    1690         LD   DE,#0000        ; X-Koordinate laden
A322            1700 X_POS:  EQU  $-2             ; Argument v. LD DE,....
A324  210000    1710         LD   HL,#0000        ; Y-Koordinate laden
A325            1720 Y_POS:  EQU  $-2             ; Argument v. LD HL,....
A327  CDC0BB    1730         CALL MOVE            ; Grafikcursor setzen
A32A  3E00      1740         LD   A,#00           ; Grafikpen laden
A32B            1750 GRAPEN: EQU  $-1             ; Argument v. LD A,..
A32C  CDDEBB    1760         CALL SGRPEN          ; Grafikpen setzen
A32F  3E00      1770         LD   A,#00           ; Grafikpaper laden
A330            1780 GRAPAP: EQU  $-1             ; Argument v. LD A,..
A331  C3E4BB    1790         JP   SGRPAP          ; Grafikpaper setzen

Pass 2 errors: 00

Table used:    706  from  1000
```

Programm 9.1: Assemblerlisting der Hardcopy-Routine (Forts.)

```
64000 ´ BASIC-Loader für Grafikhardcopy
64010 IF HIMEM>41471 THEN MEMORY &A1FF
64020 IF PEEK(&A333)=&BB THEN RETURN
64030 RESTORE 65000
64040 zeile=0
64050 FOR n%=&A200 TO &A33F STEP 16
64060     zeile=zeile+1
64070     summe%=0
64080     FOR n1%=0 TO 15
64090         READ byte$
64100         byte%=VAL("&"+byte$)
64110         POKE n%+n1%,byte%
64120         summe%=summe%+byte%
64130     NEXT
64140     READ check$
64150     IF VAL("&"+check$)=summe% THEN 64180
64160        PRINT" Ladefehler der Grafikhardcopy !!!"
64170        PRINT" Fehler in der";zeile;". DATA-Zeile":END
64180 NEXT
64190 CALL &A200
64200 RETURN
65000 ´ DATA´s für die Grafikhardcopy
65001 DATA 01,09,A2,21,14,A2,C3,D1,BC,0E,A2,C3,18,A2,4B,4F,069A
65002 DATA 50,49,C5,00,00,00,00,00,CD,C9,A2,CD,11,BC,3C,47,05B3
65003 DATA 3E,08,0F,10,FD,32,74,A2,CD,BA,BB,3E,1B,CD,9D,A2,0751
65004 DATA 3E,31,CD,9D,A2,21,8F,01,CD,B0,A2,3E,1B,CD,9D,A2,07B0
65005 DATA 3E,4C,CD,9D,A2,AF,CD,9D,A2,3E,03,CD,9D,A2,06,80,0824
65006 DATA AF,CD,9D,A2,10,FA,11,00,00,E5,06,07,0E,00,CD,BB,065E
65007 DATA A2,CB,11,2B,2B,10,F7,79,E1,E5,01,08,00,ED,42,30,0682
65008 DATA 02,E6,78,06,00,CD,9D,A2,13,10,FA,B7,21,7F,02,ED,06D5
65009 DATA 52,E1,F2,59,A2,B7,01,0E,00,ED,42,30,AB,3E,1B,CD,0716
65010 DATA 9D,A2,3E,32,CD,9D,A2,CD,B0,A2,C3,03,A3,4F,CD,1B,087A
65011 DATA BB,FE,FC,CA,03,A3,CD,2E,BD,38,F3,79,CD,2B,BD,C9,09FF
65012 DATA 3E,0D,CD,9D,A2,3E,0A,CD,9D,A2,C9,C5,D5,E5,CD,F0,09B0
65013 DATA BB,B7,28,01,37,E1,D1,C1,C9,CD,C6,BB,ED,53,22,A3,0961
65014 DATA 22,25,A3,CD,CC,BB,ED,53,07,A3,22,0A,A3,CD,D5,BB,0854
65015 DATA 22,10,A3,ED,53,13,A3,CD,D8,BB,22,19,A3,ED,53,1C,0765
65016 DATA A3,CD,E1,BB,32,2B,A3,CD,E7,BB,32,30,A3,E1,ED,73,09C1
65017 DATA 04,A3,E9,31,F2,BF,11,00,00,21,00,00,CD,C9,BB,11,0606
65018 DATA 00,00,21,00,00,CD,CF,BB,11,00,00,21,00,00,CD,D2,0449
65019 DATA BB,11,00,00,21,00,00,CD,C0,BB,3E,00,CD,DE,BB,3E,0617
65020 DATA 00,C3,E4,BB,00,00,00,00,00,00,00,00,00,00,00,00,0262
```

Programm 9.2: BASIC-Lader

PROGRAMMBESCHREIBUNG DES KOPIE-BEFEHLS

Das Assemblerlisting der Hardcopy-Routine beginnt mit der Definition der Labels (Marken).

Das Label ERWEIT hat als Wert die Adresse &BCD1 der ROM-Routine für die Befehlseinbindung. Näheres dazu im Kapitel über Befehlserweiterungen.

GETKEY fragt bei Aufruf der Adresse &BB1B die Tastatur einmalig ab und liefert den Wert der gedrückten Taste im Akku zurück.

Die Labels zwischen den Zeilen 30 und 160 sind Grafik-Routinen und wurden im Buch bereits besprochen.

Das Label GTMODE entspricht der Adresse &BC11 und liefert bei Aufruf im Akku den aktuellen MODE zurück (0,1,2).

TESTPR prüft bei Aufruf, ob der Drucker bereit ist, ein Zeichen zu empfangen. Der Wert von TESTPR ist die Adresse &BD2E.

SENDPR entspricht einer Routine, die versucht, das im Akku stehende Zeichen auf den Drucker auszugeben. Ist der Drucker nicht bereit es aufzunehmen, so kehrt die Routine nach ca. 0,4 Sekunden mit zurückgesetztem Carry-Flag wieder zurück. SENDPR entspricht der Adresse &BD2B.

In den Zeilen 210 bis 240 werden Konstanten für die Tastaturabfrage und die Druckerausgabe festgelegt.

Die Zeile 270 gibt dem Assembler zu verstehen, daß der erzeugte Maschinencode ab der Adresse &A200 abgelegt werden soll. Dadurch ist ein Umschreiben der Routine mit der Änderung der ORG-Adresse erledigt.

Von der Zeile 290 bis zu 350 stehen die Befehle, der Name und der reservierte Speicherplatz für die Einbindung der Befehlserweiterung. Das BC-Register wird zum Einbinden des Befehls mit der Adresse der Sprungtabelle und HL mit der Adresse der vier freien Bytes geladen. Dann wird ERWEIT zur Initialisierung des Befehls aufgerufen und über den RET-Befehl der ERWEIT-Routine ins BASIC zurückgekehrt.

KOPIE-BEFEHL

Wird im BASIC dieser Befehl gegeben, so wird über den in Zeile 330 stehenden Sprungbefehl JP KOPIE die Maschinen-Routine aufgerufen.

Dort wird zuerst einmal mit dem Aufruf von GRSTO der Zustand des Grafiksystems des CPC gespeichert. Das geschieht, damit beim Verlassen der Routine die Eintrittsbedingungen wieder hergestellt werden können, also der Anwender nicht etwa den Koordinatenursprung, die Grafikcursor-Position, die Grafikfensterbreite und -höhe, die Farbe des Grafikstifts und die Farbe des Grafikhintergrunds nach einer Hardcopy wieder herstellen muß.

Die folgenden Befehle errechnen aus dem Bildschirmmodus den, vom Modus abhängigen, Unterschied der X-Koordinaten benachbarter Bildpunkte. Dazu wird der MODE mit GTMODE in den Akku geholt und wegen des minde-

stens einfachen Durchlaufs einer folgenden Schleife um 1 erhöht. Dieser Wert wird in das Zählregister B geladen, und der Akku wird mit dem zweifachen maximalen Unterschied, also mit 8, geladen. In der gerade angesprochenen Schleife wird nun der Akkuinhalt mit einer Rechtsrotation halbiert. Dann wird das Zählregister dekrementiert und, wenn es noch nicht 0 ist, zum Schleifenanfang gesprungen. Durch diese Schleife wird der Ausdruck

Akku = 8 / (2* (MODE+1))

berechnet. Der dadurch erhaltene Wert wird unter der Adresse STEP gespeichert. Dort stellt er den Operanden eines LD B,x-Befehls dar.

Nach der Berechnung dieses Wertes wird das Grafiksystem des CPC mit GRINIT zurückgesetzt. D. h. der Koordinatenursprung liegt bei (0,0), und der Grafik-Cursor liegt im Ursprung. Ferner umfaßt das Grafikfenster den kompletten Bildschirm, die Farbe des Grafikstifts ist 1, und die des Grafikhintergrunds 0.

Als nächstes wird der Drucker mit der Codesequenz ESC+"1" auf einen Zeilenvorschub von 7/72 Inch eingestellt. Diese Steuerzeichenfolge ist den EPSON-Druckern und den meisten kompatiblen, also auch dem Schneider-Drucker, geläufig.

Vor dem Eintritt in die Hauptschleife wird das HL-Register noch mit 399, also der maximalen Y-Koordinate, geladen.

In der Hauptschleife wird zuerst eine Steuerzeichenfolge an den Drucker gesendet, die diesen auf eine Bit-Image-Darstellung mit 768 Punkten nebeneinander umstellt. Die Codefolge dafür lautet ESC/"L"/0/3. Die Darstellung wird auf 768 Punkte eingestellt, da der CPC nicht über einen 8-Bit-Druckerport verfügt und damit nicht die Codefolge ESC/"L"/128/2 ausgeben kann. Die überschüssigen 128 Punkte in einer Druckzeile werden durch die Ausgabe von 128 Nullbytes kompensiert, dadurch befindet sich die Hardcopy etwa in der Mitte des Druckpapiers, was einen besseren optischen Eindruck gibt.

Die Ausgabe der 128 Nullbytes erfolgt über die Schleife in den Zeilen 620 bis 640. Davor wird das Register B mit 128, also der Anzahl der auszugebenden Bytes, geladen. In der Schleife wird der Akku gelöscht und mit PRCHR ausgegeben. Mit dem DJNZ-Befehl wird, solange B nicht 0 ist, zum Schleifenanfang gesprungen.

Da mit der Hardcopy nicht nur oben, sondern auch links angefangen werden soll, wird DE mit 0 geladen, bevor die X-Schleife betreten wird.

In der X_LOOP genannten Schleife der X-Koordinaten wird erst einmal der Y-Wert auf den Stapel gerettet, die Anzahl der zu testenden Punkte, nämlich

7, in B und der Anfangswert des auszugebenden Zeichens, nämlich 0, in C geladen.

Mit diesen Werten wird eine kleine Schleife betreten, die die Farbe von sieben untereinanderliegenden Bildpunkten prüft. Dazu wird das Unterprogramm TESTXY aufgerufen, das die Registerpaare rettet, TEST aufruft und, wenn der Punkt auf eine Farbe ungleich 0 gesetzt ist, das Carry-Flag setzt. Sonst wird das Carry-Flag zurückgesetzt. Das zurückerhaltene Carry-Flag wird von rechts nach links um eine Bitposition in das Register C eingeschoben. Dann wird die Y-Koordinate in HL um 2 vermindert. Dies alles wird siebenmal ausgeführt, bevor in dem Register C das auszugebende Zeichen entstanden ist.

Der Befehl nach der kleinen Schleife holt das HL-Register vom Stapel und rettet es gleich wieder dorthin. Das BC-Register wird mit 8 geladen und von der Y-Koordinate in HL subtrahiert. War der Wert in HL kleiner als 8, so wird dadurch das Carry-Flag gesetzt, und es zeigt an, daß es sich um die letzten vier Bildzeilen handelt. Wenn dies zutrifft, so werden mit dem Befehl AND %11110001 die untersten drei Bits gelöscht.

Das Byte wird in einer Schleife so oft ausgegeben, wie es im jeweiligen MODE nötig ist. Dadurch kann Zeit gespart werden, da die Erstellung des Bytes nicht (z. B. im MODE 0 viermal) wiederholt werden muß, sondern das Byte nur wiederholt ausgegeben wird. Dazu wird B mit dem Stepwert geladen und in der Schleife STEPLP das Byte ausgegeben, DE inkrementiert, und das wird so oft, wie der Stepwert angibt, wiederholt.

Nach der Ausgabe wird das Carry-Flag gelöscht und HL mit der maximalen X-Koordinate geladen. Die aktuelle X-Koordinate steht in DE und wird von HL abgezogen. Ist DE größer als HL, so wird das Vorzeichen-Flag gesetzt, d. h. das Ergebnis ist negativ. Das HL-Register, also die Y-Koordinate, wird vom Stapel geholt, und wenn das vorherige Ergebnis positiv war, wird die nächste X-Koordinate in Angriff genommen.

Ansonsten wird das Carry-Flag gelöscht und BC mit 14 geladen. Dieser Wert entspricht sieben – ausgegebenen – Y-Zeilen. BC wird dann von HL, dem Y-Wert, subtrahiert, und wenn das Ergebnis größer als 0 ist, also noch nicht Y-Koordinate 0 erreicht wurde, wird zu NXTLIN gesprungen, um die nächste Druckerzeile auszugeben.

Ist die Routine fertig, dann wird der Drucker mit ESC+"2" auf normalen Zeilenvorschub eingestellt, ein Zeilenvorschub ausgegeben und die Routine über die Routine GRREST in Richtung BASIC verlassen.

PRCHR ist die Routine, die die Prüfung auf Abbruch durch die ESC-Taste übernimmt und das im Akku überreichte Zeichen auf den Drucker ausgibt.

Zuerst wird das Zeichen in das C-Register gerettet, GETKEY aufgerufen und mit der ESC-Taste verglichen. Wurde ESC gedrückt, so wird die Hardcopy-Routine über GRREST verlassen. Ansonsten wird der Drucker geprüft, ob er ein Zeichen empfangen kann. Wenn nicht, so springt die Routine wieder zur Prüfung der ESC-Taste. Ist der Drucker zum Empfang von Zeichen bereit, wird der Akku aus dem Register C geladen und mit SENDPR ausgegeben. Nach der Ausgabe erfolgt ein Rücksprung zur aufrufenden Routine.

Die Routine PRCRLF gibt nacheinander einen Wagenrücklauf und einen Zeilenvorschub zum Drucker aus, bevor der Rücksprung erfolgt.

TESTXY ist eine Routine, die als erstes die Registerpaare BC, DE und HL auf den Stapel rettet und die ROM-Routine TEST aufruft. Der zurückgelieferte Wert wird ODER-verknüpft. Mit dieser Verknüpfung wird zweierlei erreicht: Einmal wird das Carry-Flag gelöscht und zum zweiten der Akku auf 0 geprüft. Ist der Farbwert 0, so werden die Registerpaare wieder geholt, und die Routine wird verlassen. Ist der Farbwert ungleich 0, so wird vor dem Holen der Register und dem Rücksprung noch das Carry-Flag gesetzt.

Die Routine GRSTO wird am Anfang des Programms aufgerufen. Sie holt sich über die XY_POS-Routine erst einmal die Koordinaten des Grafik-Cursors und speichert sie unter den Adressen X_POS und Y_POS. Mit GETORG wird die Lage des Koordinatenursprungs geholt und unter den Adressen ORG_X und ORG_Y gespeichert. Die X-Koordinaten des Grafikfensters werden mit GETX12 geholt und unter WIN_X1 und WIN_X2 gespeichert. Die Speicherung der Y-Koordinaten wird unter den Adressen WIN_Y1 und WIN_Y2 durchgeführt. Die nötigen Koordinaten werden mit der Routine GETY12 geholt. Die beiden vorletzten Vorgänge in dieser Routine sind das Holen des Grafikstifts und seine Speicherung unter GRAPEN sowie die Speicherung des mit GGRPAP geholten Grafikhintergrunds unter der Adresse GRAPAP. Zum Schluß wird die Rückkehradresse vom Stapel ins HL-Register geladen, der Stapelzähler unter der Adresse STPTR gespeichert und mit einem Sprung an die in HL stehende Adresse zurückgekehrt.

Die letzte Routine GRREST wirkt umgekehrt zu GRSTO, d. h. der Stapelzähler wird mit dem von GRSTO gespeicherten Wert geladen. Dann werden DE und HL mit dem Koordinatenursprung geladen, und die SETORG-Routine wird aufgerufen, die den Origin setzt. Als nächstes werden DE und HL mit den seitlichen Begrenzungen geladen, und die Grafikfensterbreite wird mit SETX12 gesetzt. Die Grafikfensterhöhe wird mit der SETY12-Routine gesetzt, nachdem DE und HL mit den unter WIN_Y1 und WIN_Y2 gespeicherten Daten geladen wurden. Schließlich wird noch das DE-Register mit der X- und HL mit der Y-Koordinate geladen und der Grafik-Cursor mit MOVE ge-

setzt. Die vorletzte Anweisung lädt den Akku mit der Grafikstiftfarbe und setzt diese entsprechend. Zum Schluß wird der Akku mit der Grafikhintergrundfarbe geladen, und diese wird mit SGRPAP gesetzt. Die Rückkehr erfolgt über den Return-Befehl der SGRPAP-Routine.

Kapitel 10
Besondere Grafikeigenschaften des CPC 664 und des CPC 6128

Im letzten Kapitel möchten wir aus Gründen der besseren Übersicht nochmals die Unterschiede zwischen dem CPC 464, CPC 664 und CPC 6128 in bezug auf unser Thema Grafik zusammenfassen.

Neben den äußeren Unterschieden (Floppylaufwerk/Kassettenrecorder) und der Geräumigkeit des Speichers (64 KB/128 KB), die nur mittelbar die Grafikanwendung eines CPC-Rechners unterstützen, muß insbesondere auf eine wichtige Befehlsänderung und auf die neuen Befehle hingewiesen werden.

INK-MODUS BEI GRAFIKBEFEHLEN

Die Befehle DRAW, DRAWR, MOVE, MOVER, PLOT und PLOTR wurden mit einem begrüßenswerten Zusatz ausgestattet. Bei diesen Befehlen kann sofort der sogenannte INK-Modus angegeben werden, ohne ihn umständlich über CHR$-Sequenzen zu programmieren. Dieses Thema wurde im letzten Teil von Kapitel 5 besprochen.

FILL

Eine weitere Neuerung bei den Rechnern 664 und 6128 ist der FILL-Befehl, der bei modernen BASIC-Versionen durchaus schon zum Standard zählt und sicherlich beim 464 sehnlichst vermißt wurde und wird. Er erlaubt das Ausfüllen komplett umrandeter Flächen mit einer Farbe und erspart so lästige Programmierarbeit und wertvolle Rechenzeit bei grafischen Anwendungen, die auf farbige Flächen zurückgreifen.

Wichtig ist zu erwähnen, daß der Rand für die einzufärbende Fläche entweder die Farbe haben muß, mit der die Fläche gefüllt werden soll, oder die Farbe des aktuellen Grafikstiftes.

GRAPHICS PAPER UND GRAPHICS PEN

Diese Befehle ermöglichen es bei den beiden neueren Versionen, auch dem Grafikhintergrund und dem Zeichenstift eine eigene Farbe zuzuordnen. Dies unterstützt besonders die Anwendung des Transparent-Modus.

MASK

Bei Liniendiagrammen und anderen grafischen Darstellungen, die auf viele Linien zurückgreifen, wobei den Linien unterschiedliche Bedeutung zugemessen wird, dürfte sicherlich der MASK-Befehl eine wertvolle Hilfe sein, da er schraffierte Linien mit bis zu 255 unterschiedlichen Schraffuren ermöglicht. Durch Eingabe einer 1-Byte-Maske kann der Anwender selbst das Aussehen seiner Linie bestimmen.

FRAME

Besonders bei der Arbeit mit Maschinenprogrammen im Bereich Grafik tritt beim CPC 464 manchmal ein störendes Flackern auf. Es kann mit dem Befehl CALL &BD19 behoben werden. Bei den beiden neueren Rechnern ist dies noch einfacher geworden: Das Zauberwort heißt FRAME.

COPYCHR$

Als letztes sei noch die Möglichkeit erwähnt, Grafikteile von einem Bildschirmbereich in einen anderen zu kopieren. Hier leistet der neue Befehl COPYCHR$ wertvolle Hilfe.

Die Firma Schneider hat mit den neuen Computern 664 und 6128 sicherlich einen Schritt nach vorne gemacht. Das Thema unseres Buches, die Grafik, wurde bei den Neuerungen jedoch recht stiefmütterlich behandelt. Sicherlich sind die oben genannten Ergänzungen in manchen Fällen eine wertvolle Hilfe. Andere Rechner, z. B. der C128 von Commodore, bieten jedoch auch Befehle für geometrische Figuren wie Rechtecke und Blöcke, die sogar um einen beliebigen Winkel gedreht werden können.

Deshalb ist die in Kapitel 2 beschriebene Ergänzung – besonders in Form von BASIC-Befehlen – auch in vielen Fällen zur sinnvollen Anwendung von Grafik noch nötig.

Stichwortverzeichnis

Die SYBEX-Bibliothek

Schneider

SCHNEIDER CPC 464: MEIN ERSTES BASIC PROGRAMM
von Rodnay Zaks – zahlreiche farbige Illustrationen und viele Diagramme helfen Ihnen, auf spielerische Weise in BASIC zu programmieren; ohne Vorkenntnisse nutzbar. 208 Seiten, zahl. farb. Abb., Best.-Nr. **3096** (1985)

MEIN SCHNEIDER CPC
von Norbert und Christoph Hesselmann – von der Hardware-Umgebung über ein umfangreiches BASIC-Lexikon, Grafikentwurf und Musikerzeugung, Mikroprozessorentechnik und Maschinensprache, Arbeiten mit dem Disketten-Laufwerk bis hin zu den Betriebssystemen AMSDOS und CP/M 2.2. 376 Seiten, 124 Abbildungen, Best.-Nr.: **3602** (1985)

ARBEITEN MIT DEM SCHNEIDER CPC
von Hans Lorenz Schneider – eine umfassende und didaktisch aufbereitete Arbeitshilfe für Anfänger, aber auch Fortgeschrittene finden ein Bündel von Tips und Tricks. 288 Seiten, 113 Abbildungen, Best.-Nr.: **3603** (1985)

SCHNEIDER CPC 464 ASSEMBLER KURS
Reihe MISTER MICRO – Das Buch führt Sie schrittweise in die Programmierung des Z80 ein und vermittelt Ihnen Befehlssatz des Prozessors wie Adressierungsarten. 232 Seiten, mit Abbildungen, Buch und Kassette, Best.-Nr.: **3412** (1985)

SCHNEIDER CPC STARTEXTER
von Reinhold Krumscheid – Das Textverarbeitungs-Programm der Spitzenklasse auch für den Schneider CPC 464/664/6128. Mit Profi-Möglichkeiten zum kleinen Preis. Diskette + Trainingsbuch, Best.-Nr. **3416** (1986)

Assembler

PROGRAMMIERUNG DES Z80
von Rodnay Zaks – ein umfassendes Nachschlagewerk zum Z80-Mikroprozessor – jetzt in einer durch Lösungen ergänzten Ausgabe. 2., erweiterte Ausgabe. 640 Seiten, 176 Abbildungen, Best.-Nr.: **3099** (1985)



Fordern Sie ein Gesamtverzeichnis unserer Verlagsproduktion an:

SYBEX-VERLAG GmbH
Vogelsanger Weg 111
4000 Düsseldorf 30
Tel.: (02 11) 61 80 2-0
Telex: 8 588 163

SYBEX INC.
2344 Sixth Street
Berkeley, CA 94710, USA
Tel.: (415) 848-8233
Telex: 287 639 SYBEX UR

SYBEX
6–8, Impasse du Curé
75018 Paris
Tel.: 1/203-95-95
Telex: 211.801 f

SYBEX

Das Schneider CPC Grafikbuch

Alle Grafikmöglichkeiten der CPC-Computer von Schneider auf einen Blick. Schritt für Schritt führt der Autor Sie vom Aufbau der Grafik bis hin zum Druck.

Nach einer Einführung (Hardware, Betriebssystem-Routinen, vorhandene Grafikbefehle) werden Ihnen ergänzende Grafikbefehle vorgestellt. Diese werden zunächst als BASIC-Unterprogramme erläutert und dann als Maschinenprogramm übertragen – mit vielen Anwendungsbeispielen. Das Kapitel „Grafiken mit dem Joystick erstellen" hilft Ihnen, die Erweiterungen praktisch anzuwenden.

Ein weiteres Kapitel widmet der Autor verschiedenen Diagrammformen, wobei die Diagramme zwei- und auch dreidimensional dargestellt werden. Auch künstlerische Grafiken kommen nicht zu kurz, wobei laufende Farben und Transparent-Modus im Vordergrund stehen.

Das Zusammenspiel zwischen Grafik und Datenträger (Speichern, Laden und Mischen) ist für Ihre Grafikentwicklungen ebenso wichtig wie Sprites und ein Hardcopy-Programm, um die erstellte Grafik zu Papier zu bringen.

Ein umfangreiches Stichwortverzeichnis und zahlreiche Abbildungen helfen Ihnen, die vielfältigen Grafikmöglichkeiten Ihres Schneider CPC vom Start weg sicher zu nutzen.

Über den Autor

Hans Lorenz Schneider studierte Informatik und schrieb seine Diplomarbeit am bereits legendären PET. Seit 1980 führt er ein Software-Haus, in dem er individuelle Programme für Personal-Computer anbietet. Seine Kenntnisse und Erfahrungen publiziert H. L. Schneider seit 1982 in Artikeln und Büchern, u. a. im SYBEX-Titel „Arbeiten mit dem Schneider CPC".

ISBN 3-88745-611-4

DM 48,—
sfr 44,20
öS 374,—



3611 SYBEX

Das Schneider CPC Grafikbuch

Schneider

AMSTRAD CPC

OCR

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA



https://acpc.me/